MICROCOMPUTADOR

CURSO PRÁTICO

rioGráfica

MICROCOMPUTADOR CURSO PRÁTICO

rioGráfica

Diretoria Executiva
Oscar Neves
Filipe Zander
Nilo Sérgio de Almeida
Danilo Esteves Costa

GRUPO EDUCAÇÃO E CULTURA

Planejamento e Comercialização

Diretor
Roberto Combochi
Gerente de Marketing
Jaime Rodrigues
Gerente de Produto
José Mauro Porto

Editorial

Diretor
João J. Noro
Editor-chefe
Maurício Rittner
Administração Editorial
Heitor de Souza Paixão

Editor
Jesse Navarro
Revisores
Isis Augusta Loyolla
Cacilda Guerra
Luiz Vicente Vieira Filho
Marcos Domingos Agathão
Maria da Graça Mendonça Couto
Diretor de Arte
Aníbal Monteiro
Chefe de Arte
Bonifácio Duardes Miranda
Chefe de Estúdio
José Yuji Kuribayashi
Assistente de Arte
Miguel Luis Escámez Simón

Colaboradores
Editores-assistentes:
Isa Mara Lando
Luiz Carlos Pizarro Marin
Texto:
Geraldo Carlos Nascimento
Preparação de Texto:
Luiz Carlos Cardoso
Tradução:
Cláudio Marcondes
Maria Clara Cescato
Patrícia de Paiva e Castro
Regina Amarante
Regina Carmo R. Silva
Pesquisa Iconográfica:
Stella Maria Quentel
Fotos Nacionais:
Fernado Scavone
Salvador de Rosa
Assessoria:
Derval Junqueira de Aquino
Lauro de Lauro
Luiz Henrique Alayon

Composição: ERG Ltda. e AM Produções Gráficas Ltda.
Impressão: JBIG

MICROCOMPUTADOR
CURSO PRÁTICO

EDITORA
rioGráfica
LAZER, CULTURA E INFORMAÇÃO

Apresentação

Em poucos anos, o computador perdeu a aura de mistério e tornou-se parte integrante de nosso ambiente diário. Sua tecnologia está presente em casa, no banco, no escritório e até no pulso das pessoas, em forma de relógio digital.

Diante disso, há cada vez mais pessoas recebendo treinamento especializado em computação. Assim acompanham os rápidos e constantes progressos no setor. A maioria dos aficionados procura ir além do mero aprendizado de tarefas, buscando uma compreensão geral dos sistemas computacionais e de seus recursos. Muitos têm em vista alargar conhecimentos ou oportunidades de trabalho.

É o que pretende esta obra: não apenas dar elementos para se programar e operar um micro, mas também mostrar como ele pode ser bem utilizado na vida diária e profissional. MICROCOMPUTADOR — CURSO PRÁTICO destina-se tanto ao jovem a caminho da profissionalização como ao programador que pretende desenvolver-se. Dirige-se, sobretudo, ao usuário desejoso de mais informação e maior universo de opções.

A obra, além de apresentar meios para programação e análise dos sistemas básicos, oferece uma visão ampla dos micros e periféricos existentes, com indicação precisa de seus princípios de funcionamento. Examina ainda as aplicações do computador, softwares que as tornam possíveis e as principais linguagens em uso, sem esquecer a necessidade de uma base teórica, nem o lado humano e empresarial da microinformática.

Com tal diversidade e amplitude de temas, MICROCOMPUTADOR — CURSO PRÁTICO espera que você não só tire o máximo do micro como obtenha dados essenciais dessa tecnologia que está mudando a face da sociedade em que vivemos. Mudar junto com ela é fundamental para termos, hoje, o futuro nas mãos.

Os editores

Conheça seu curso

As listagens constantes da obra foram testadas, mantendo a preocupação de abranger as principais linhas de equipamentos. As seções podem ser facilmente identificadas pela tarja de cor.

Aplicações — Oferece uma visão ampla das aplicações do micro nas mais diversas áreas de atividade.
Periféricos — Examina grande variedade de equipamentos disponíveis, explicando seus princípios de funcionamento e sua utilização.
Software — Mostra as características de programas de vários tipos e analisa detalhes dos mais importantes "pacotes" disponíveis.
Raio X — Desvenda a estrutura interna de hardwares e periféricos, dando as respectivas fichas técnicas.
Ciência da computação — Lições de lógica computacional, que vão desde os fundamentos teóricos até sua aplicação prática na arquitetura interna do micro.
Oficina — Série de artigos bem ilustrados que mostra como fazer, em casa, conexões, pequenos consertos e a manutenção geral do equipamento.
Linguagens — Cursos de LOGO e COBOL e noções de ASSEMBLER, como linguagens alternativas ao BASIC; este já foi analisado na coleção MICROCOMPUTADOR — CURSO BÁSICO e no Curso de Programação BASIC dela extraído (publicações da Editora Rio Gráfica).
Projetos de programas — Apresenta propostas e listagens originais, desenvolvidas em linguagem BASIC, que resultam na geração de jogos e de úteis softwares aplicativos.
Técnicas de programação — Orienta o usuário a desenvolver programas mais eficientes em BASIC, permitindo melhores resultados.
Perfil — Pessoas e empresas, do Brasil ou do exterior, que se destacaram na área de microinformática.
Registre — Sugestões práticas que resolvem problemas inesperados; aparecem em quadros ao longo das seções.
Jogos — Programas prontos para utilização, formando um volume à parte.

PAUL CHAVE

A MÁQUINA DE HOJE

O microcomputador, em 1978, não passava de brincadeira entre amigos inventivos. Mas graças ao barateamento dos componentes tornou-se uma indústria milionária e integra o cotidiano de milhões de pessoas.

Fruto da calculadora, da máquina de escrever e da televisão e tendo como antepassado distante o tear automático, que também usa o sistema binário para urdir, o computador pode ser considerado uma síntese do desenvolvimento tecnológico. Em suas minúsculas pastilhas de silício — chips —, a capacidade de armazenamento de informação continua aumentando, o que reduz o preço dos aparelhos cada vez mais.

O barateamento possibilitado pelos chips trouxe como conseqüência, a partir dos anos 70, a extraordinária popularização dos microcomputadores. A tendência é de que estes passem a ser tão comuns nas residências quanto o televisor e o telefone. Serão usados para ligar aparelhos domésticos, controlar orçamentos, redigir cartas ou, simplesmente, entreter a família com jogos de perícia, lógica ou estratégia.

Os computadores dependem dos programas. Um computador sem programa assemelha-se a um projetor sem filme ou a uma vitrola sem disco — não funciona. Os programas são conjuntos de instruções para que o computador realize determinadas tarefas. Vêm armazenados em fitas cassete, disquetes flexíveis ou cartuchos. No momento do uso, passa-se o programa para a memória do computador, que o executa na resolução de tarefas ou de problemas.

O programa permite processar (isto é, introduzir, armazenar, classificar e alterar) dados numéricos, alfabéticos ou gráficos. Impressos, os

Avanço e difusão
Em menos de uma década, os computadores — inclusive os micros — tornaram-se parte integrante da vida de milhões de pessoas em todo o mundo. Suas aplicações, hoje, são praticamente irrestritas.

dados processados aparecem sob forma de relatórios, planilhas, gráficos ou desenhos.

Os programas são comprados prontos (pacotes) ou desenvolvidos pelo próprio usuário. Com o micro, uma só pessoa exerce as funções de analista, programador e usuário.

Há entusiastas amadores que têm prazer em planejar e codificar seus próprios programas. Para isso, usam a popular linguagem BASIC ou as linguagens de alto nível, isto é, próximas da linguagem humana.

A programação não só produz resultados práticos como constitui um exercício para o raciocínio e a criatividade, pela manipulação de um conjunto de regras lógicas.

Os sistemas computacionais seriam incompletos se não contassem com os equipamentos periféricos, que, acoplados aos micros, aumentam muito sua possibilidade de ação. A tela de vídeo e a impressora constituem os periféricos mais importantes, mas há também o joystick, os tabletes digitalizadores, a caneta óptica e o mouse.

Na área comercial e industrial, o micro popularizou-se nas pequenas e médias empresas, que em geral utilizam programas prontos para suas principais atividades contábeis e financeiras, tais como folhas de pagamento, controle de estoque, balanço ou previsão orçamentária.

Os micros também prosperam nos escritórios, que tendem à automação. Com grande freqüência as empresas modernas utilizam o processamento eletrônico de textos para os serviços de mala direta e de redação de cartas personalizadas. Os jornais, por exemplo, beneficiam-se muito dos terminais em que os repórteres digitam e corrigem suas matérias; elas ficam armazenadas numa memória central, e podem ser depois revisadas e editadas com facilidade.

Tendência semelhante se verifica entre profissionais como médicos, engenheiros, arquitetos, advogados, empresários rurais. Em todos esses e em muitos outros ramos de atividade, os micros prestam serviços de gerenciamento de dados, substituindo com vantagem os arquivos de fichas, que são volumosos e, quase sempre, muito difíceis de pesquisar.

A telecomunicação, interligando micros e computadores de grande porte por meio de linhas telefônicas, abre novos horizontes para a sociedade informatizada. O acesso dos usuários de micros a grandes bancos de dados eletrônicos configura, cada vez mais, uma democratização da informação.

Na educação, o micro é um valioso auxiliar quando acoplado a equipamentos periféricos sofisticados, como vídeos e discos a laser. A instrução auxiliada por computador já é uma realidade, sobretudo no treinamento profissional oferecido por empresas. E, na área da educação infantil, a linguagem LOGO revelou-se instrumento valioso para o desenvolvimento do raciocínio e da metodologia científica.

A arte com auxílio do computador encontra cada vez mais adeptos. O micro oferece recursos para a fácil manipulação de linhas, formas e cores. Faculta, assim, a expressão artística mesmo a pessoas que não desenvolveram habilidade necessária para desenhar ou pintar. A isso se acrescenta a considerável vantagem de o micro ser dotado de recursos de memória que permitem armazenar as imagens, fundi-las com outras ou mesmo modificá-las, como se fossem dados numéricos ou alfabéticos.

O microcomputador-instrumento musical não estava na mira de seus criadores, mas, com recursos elementares para produzir os sons da escala comum, ele é programável para emitir esses sons em determinada seqüência. Com procedimentos simples conseguem-se obter a harmonia e o contraponto, que são matematicamente codificáveis.

Uma aplicação está revolucionando o processo de composição musical: o uso de sintetizadores que acoplam os recursos sonoros de instrumentos sofisticados aos recursos matemáticos e lógicos do micro.

A popularidade dos microcomputadores resultou também de sua utilização para jogos. Unindo recursos de computação, imagem e som, o micro simula os perigos de uma guerra nas estrelas ou as emoções de uma corrida de Fórmula 1. Sem falar em sua capacidade para se transformar num parceiro inteligente, capaz de defrontar-se com um oponente humano num jogo de estratégia e raciocínio. O exemplo clássico é o xadrez.

Juntos, o hardware (os equipamentos) e o software (os programas) abrem possibilidades quase infinitas, não só para as aplicações práticas e científicas como também para as educacionais e recreativas. Os micros funcionam como extensões do cérebro e dos órgãos sensoriais e motores do homem. Possibilitam boa parte do que a imaginação concebe.

Micro sem fio
A mais avançada tecnologia contribui para a sofisticação dos equipamentos buscando facilitar seu manuseio. O modelo portátil da Apricot, por exemplo, é dotado de microfone, que permite o reconhecimento da voz. O mouse e o teclado não se conectam ao computador por fios: o acionamento é feito por raios infravermelhos.

CÓDIGOS LÓGICOS

Um conjunto de símbolos, palavras e comandos governado por regras fixas compõe a linguagem computacional — instrumento de comunicação do homem com a máquina, cada vez mais próximo da expressão verbal humana.

Nos primórdios da computação, usava-se o código de máquina (expresso em notação binária) ou ASSEMBLER, linguagem de baixo nível muito próxima desse código. Com a evolução da informática, criaram-se as linguagens de alto nível, semelhantes à fala e ao raciocínio humanos. Nessas linguagens, uma só instrução equivale a diversos comandos primários, o que facilita a tarefa do analista/programador. Nas décadas de 60 e 70 apareceram centenas de linguagens, algumas já firmadas entre os programadores profissionais, em aplicações comerciais, educacionais e científicas, incluindo desenvolvimento de software e pesquisas de inteligência artificial.

FORTRAN
(FORmula TRANslator)
– 1956 –
A mais antiga, continua sendo utilizada na ciência, engenharia e desenvolvimento de software.

COBOL
(COmmon Business Oriented Language)
– 1960 –
Idealizada para aplicações comerciais, em que se manipulam grandes arquivos de dados.

ALGOL
(ALGOrithimic Language)
– 1960 –
Uma notação algébrica completa para solucionar problemas lógicos por meio de algoritmos.

LISP
(LISt Processor)
– 1960 –
Trabalha com listas de dados não numéricos. Utilizada em simulação, análise lingüística e inteligência artificial.

PL/1
(Programming Language/1)
– 1964 –
Semelhante à COBOL na fácil manipulação de arquivos, mas com recursos matemáticos da FORTRAN.

BASIC
(Beginner's All-purpose Symbolic Instruction Code)
– 1964 –
Projetada para fácil aprendizado e utilização, é ainda a mais popular das linguagens.

APL
(A Programming Language)
– 1967 –
Favorita dos matemáticos e estatísticos devido a seus poderosos operadores, que substituem dezenas de instruções de outras linguagens.

PROLOG
(PROgramming in LOGic)
– 1970 –
Manipula afirmações lógicas. Usada nas pesquisas de inteligência artificial, foi escolhida pelo Japão no desenvolvimento da quinta geração de computadores.

PASCAL
– 1971 –
Simples e ao mesmo tempo poderosa, exige sólida estruturação lógica dos programas.

FORTH
– 1975 –
No desenvolvimento de softwares e aplicativos, gera programas de rápida execução e economia de memória.

C
– 1978 –
Voltada para a elaboração de software avançado, desenvolveu o sistema operacional NIX e os efeitos especiais do filme *Guerra nas estrelas*.

MODULA 2
– 1980 –
Voltada para o desenvolvimento de sistemas, baseia-se em módulos independentes.

LOGO
– 1980 –
Projetada para crianças, desenvolve o raciocínio lógico, a habilidade de resolver problemas e o gosto pela matemática e computação.

ADA
– 1982 –
Extremamente poderosa, foi adotada pelo Departamento de Defesa dos Estados Unidos para todos os seus projetos.

MÉTODOS DE TRABALHO

Quem aprende a programar sozinho, recorrendo aos manuais técnicos que acompanham o microcomputador, pode enfrentar algumas limitações. Esta seção começa por esclarecer métodos realmente profissionais.

Chega-se à boa programação por meio da experimentação e da prática. Com o tempo, um principiante, que resolve dificuldades na base do entusiasmo, pode transformar-se num perito, chegando até mesmo a desenvolver qualidades fundamentais como a clareza e a abordagem direta do problema.

O progresso pessoal de um programador autodidata, porém, pode e deve ser acelerado pelo conhecimento dos erros de outros que se aventuram pelos mesmos caminhos.

A programação é um processo de solucionar problemas. Grande parte desse processo começa mentalmente, com lápis e papel na mão. E não, como se poderia pensar, escrevendo as instruções do programa. As etapas são conhecidas: uma clara exposição do problema, seguida de reexposições com precisão crescente, até que o problema seja formulado com o máximo de detalhes e exatidão.

A exposição quase sempre já contém ou implica uma solução. Embora não esteja evidente, ela requer apenas uma formulação mais bem elaborada para se tornar um método de trabalho — um programa. Somente o último estágio, resultado direto dos precedentes, envolve a codificação das instruções.

As soluções são conhecidas como *algoritmos*, ou seja, processos de computação analisados em estágios lógicos. A eficiência de um programa depende basicamente do algoritmo. E ele é avaliado em termos de correção e abrangência — duas qualidades que envolvem a habilidade teórica e prática do programador ao lidar com a amplitude previsível das condições de entrada e com a consistência de sua lógica interna. É mais fácil reconhecer a ausência de tais elementos que demonstrar sua presença, mas todo programa precisa ser avaliado quanto a eles logo no início de sua elaboração.

Além de completas e corretas, as soluções têm de ser confiáveis. Precisam não só dar conta da faixa previsível de problemas, mas também lidar com segurança quando surgem condições fora dessa faixa. Não é fácil, porém, julgar se um programa é suficientemente confiável.

A elaboração de programas confiáveis e consistentes — meta sempre perseguida — entra em conflito com uma outra necessidade do programador: obter um programa econômico. Tudo custa dinheiro, ainda que seja apenas o tempo gasto fazendo programas para diversão. Por isso, chega o momento de decidir se vale a pena continuar trabalhando num programa ''genial'' ou se é melhor abandoná-lo e iniciar um projeto mais realista.

Teste e eliminação de erros

Solucionar problemas analíticos ou lógicos na teoria é muito importante, mas não se pode perder de vista que os programas devem executar uma tarefa específica. Assim que os primeiros erros de sintaxe e lógica tiverem sido corrigidos, passa-se para os testes. Estes precisam ser planejados para cercar as partes mais vulneráveis do programa — justamente aquelas que devem ser as mais fortes.

Os testes bem-sucedidos revelam as inadequações do programa e as demonstram de maneira lógica, para que a depuração dos erros tome o menor tempo possível.

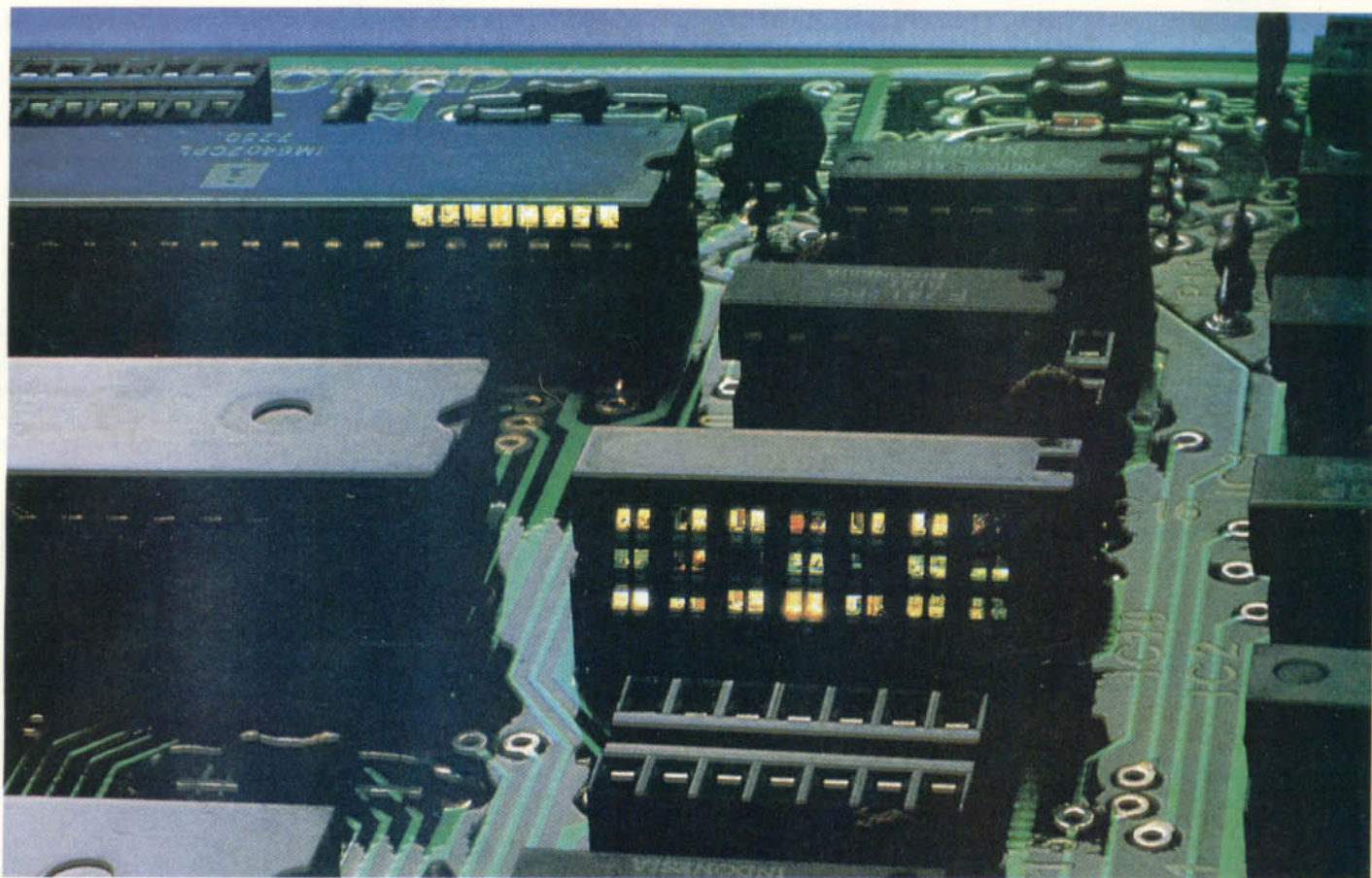
Essa depuração constitui outro processo muito importante. Ela deve ser abordada como os demais problemas a resolver: exposição, análise, algoritmo e teste.

Freqüentemente, porém, o problema é tratado como erva daninha: precisa ser destruída a qualquer custo. E isso traz conseqüências desastrosas para o programa.

Vencidas todas as etapas, a familiaridade e a satisfação combinam-se para convencer o programador de que seu programa funciona e sempre irá funcionar, além de ser um modelo de clareza. Nada mais falso.

Os programas requerem uma documentação interna, com linhas de comentários que expliquem o que se faz a cada sub-rotina, além de documentos externos, como um bom manual. Isso facilitará a manipulação do usuário, mesmo que ele seja o próprio programador.

São lições que tiveram de ser aprendidas por profissionais envolvidos com computadores de grande porte. Ignoradas por algum tempo, elas precisaram ser redescobertas pelos programadores de micros. Hoje, esses procedimentos compreendem uma ''estrutura'' de programação e uma abordagem unificada da solução de problemas — algo muito mais abrangente do que as instruções de um manual que ensina evitar o excesso de GOTO ou de WHILE-WEND.

Programas eficientes são elaborados por programadores competentes, que se fundamentam sobretudo na experiência estruturada ao longo do tempo e no raciocínio lógico.

AS FAMÍLIAS DE MICROS

Estimulados pela popularização dos microcomputadores, os projetistas de hardware de todo o mundo basearam-se nos produtos mais consagrados para criar suas próprias máquinas — os "compatíveis".

Os microcomputadores alcançaram enorme sucesso a partir do fim da década de 1970. Seus protótipos tiveram origem em laboratórios caseiros. Segundo os criadores do Apple, por exemplo, esse micro foi não só desenvolvido numa garagem como também projetado para requerer poucos recursos em sua construção.

Os fabricantes de computadores de grande porte também desenvolveram projetos de micros, mas apenas a IBM obteve sucesso, com o IBM PC (Personal Computer).

No Brasil, diversos fabricantes lançaram-se no mercado, com modelos assemelhados ao Sinclair, ao Apple, ao TRS-80 ou ao IBM PC (os "compatíveis"). Definiram-se, assim, quatro "famílias" principais.

Essas famílias distribuem-se em três faixas bem distintas, no que se refere à configuração básica, aos recursos e, conseqüentemente, ao preço. Numa primeira faixa estão os equipamentos da linha Sinclair, que custam o mesmo que um aparelho de som simples. Servem para o usuário se iniciar na computação e aplicam-se em geral a jogos e à programação em BASIC.

No estágio médio estão as linhas Apple e TRS-80, aptas para aplicações profissionais, e custando até quinze vezes mais que a linha Sinclair, conforme sua configuração.

No extremo superior está a linha IBM PC, com micros que custam até quatro vezes mais que os do segmento anterior. São equipamentos potentes, de utilização quase exclusivamente profissional, grandes recursos para armazenamento de dados e acesso aos softwares mais sofisticados.

Fator decisivo para a popularização de certas linhas foi o software disponível. Por exemplo: o Visicalc, primeira planilha eletrônica a alcançar êxito, ajudou muito na divulgação da linha Apple.

Arquitetura interna
A complexa tecnologia dos microcomputadores foi adaptada em todo o mundo. No Brasil, com modelos assemelhados ao Sinclair, ao Apple II, ao TRS-80 e ao IBM PC, formaram-se quatro famílias ou linhas de compatíveis.

Linha Sinclair

Destaca-se pelo custo relativamente baixo e pela facilidade de operação. Seus pequenos micros acoplam-se à TV doméstica e utilizam gravador de fita cassete comum para armazenar dados e programas. Por isso, são os aparelhos preferidos por muitos principiantes, e foram os primeiros micros a conquistar o mercado mundial.

Em geral, os equipamentos da linha Sinclair são usados para jogos e programação em BASIC, pois têm excelente capacidade de cálculo. Para aplicações comerciais mais extensas, vêem-se limitados por dois fatores: o teclado, que não se presta a longas tarefas de digitação, e o armazenamento de dados exclusivamente em fita, que permite acesso apenas seqüencial.

Mesmo assim, há no mercado programas que oferecem recursos básicos de planilha financeira, banco de dados e editor de texto para os equipamentos da linha Sinclair.

A engenhosidade e o pioneirismo dos experimentadores também vem operando positivamente no sentido de dotar esses micros de interfaces para diversos periféricos. No mercado internacional, eles já dispõem até mesmo de unidades de disco, modems e sintetizandores de voz. No Brasil, existe interface para impressora matricial, o que oferece uma vantajosa alternativa às impressoras térmicas.

Por estarem numa faixa de preço muito abaixo das demais famílias, os micros da linha Sinclair continuam como primeira opção a quem se inicia na informática.

COMPLEMENTOS

Interface: Circuito ou ligação que compatibiliza sinais entre itens do hardware e permite conexão entre eles.
Unidade de disco: Unidade que registra e recupera informações na superfície magnética de um disco flexível em rotação. Fica bem mais dispendiosa que o uso de gravador cassete, mas armazena mais dados e trabalha com maior velocidade.
Modem: Dispositivo que permite a um computador transmitir e receber dados por meio de uma linha telefônica.
Sintetizador de voz: Acessório que possibilita a simulação eletrônica da voz humana.

TKs e CPs
A família a que Sir Clive Sinclair deu origem tem, entre seus descendentes brasileiros, o CP 200 *(à direita)* e o TK 85 *(abaixo)*, aparelhos recomendados para quem está se iniciando na computação.

Videogames

O videogame mais difundido no mercado mundial, o Atari, torna-se um microcomputador simples quando acopla um teclado apropriado e o cartucho BASIC, ambos produzidos pela Milmar. Também da Milmar e igualmente adaptável para microcomputação é o videogame Dactari.

Transformado num micro rudimentar, o Atari apresenta grande capacidade gráfica, com recursos de cores. Funciona com RAM de 16 Kbytes e ROM de 4 Kbytes, e conta, como periféricos, com o joystick e seu potente vídeo.

A impressão dos programas, no entanto, é impossível, pois ainda não se desenvolveu a interface para impressora.

Apple e TRS-80

Dotadas de teclado mecânico e com recursos para expansão de memória e armazenamento de dados em disquetes flexíveis, essas duas linhas são usadas em larga escala em aplicações profissionais e educacionais. Contam com vasta biblioteca de software: processadores de texto, bancos de dados, planilhas e pacotes financeiros, geradores de gráficos, jogos sofisticados e controle de aparelhos domésticos. Dispõem ainda de recursos de cor e som e boa capacidade gráfica.

A linha Apple caracteriza-se por sua configuração básica, expansível, permitir que o usuário monte aos poucos um sistema completo e versátil. Seus oito slots (encaixes) permitem o acoplamento de placas para expansão de memória e inúmeros periféricos. O sucesso das linhas Apple e TRS-80 no mercado mundial reflete-se sobretudo no grande número de compatíveis de várias fabricações.

No Brasil, a TRS-80 tem se destacado pela participação pioneira em projetos nacionais de telecomunicação, que dão acesso a grandes bancos de dados remotos abertos a todos os usuários, ou de interesse específico de diversos grupos profissionais.

Nessa linha, pode-se incluir também o TRS Color, cujo compatível nacional, o CP 400 Color, dispõe de programas em cartucho que possibilitam o desenho de elaborados gráficos em cores.

O meio do campo
O JR Sysdata *(em cima)*, o CP 500 *(acima)* e o TK 2000 *(à esquerda)* integram a linha TRS-80, que atende a uma faixa intermediária dos usuários de microcomputadores.

Linha IBM PC

Os equipamentos dessa linha, de preço bem mais elevado que os demais, possuem todos os recursos dos da linha Apple num hardware mais estável e famoso por não apresentar problemas técnicos. Suas dez teclas de função agilizam a execução dos programas.

Os compatíveis com IBM PC dispõem de softwares poderosos e sofisticados, graças a sua extensa capacidade de memória. As máquinas dessa linha permitem o teleprocessamento, funcionando em rede com outros micros ou computadores de grande porte.

Também está gerando compatíveis nacionais de várias fabricações o IBM XT, que inclui unidade de disco rígido, de 5, 10 ou 20 Mbytes e tem oito slots para expansões de vários tipos (contra cinco no PC).

Por seu preço, o IBM PC caracteriza-se como um micro de uso profissional, e não doméstico.

Três em um
O Ego, primeiro microcomputador brasileiro compatível com a poderosa linha IBM PC, tem aplicação sobretudo nas áreas comercial, educacional e científica. Uma unidade central, um teclado e um monitor compõem o equipamento básico.

Família	Principais compatíveis brasileiros	Microprocessador	Sistema operacional	Memória			Vídeo incluído na configuração básica	Cor	Som	Teclado	Armazenamento de dados	Periféricos acoplados
				ROM	RAM Básica	RAM Expandida						
Sinclair	TK 83, TK 85 CP 200, Ringo	Z80A	P1	8 ou 10K	2, 16 ou 48K	Até 48K em módulos de 16K	Não	Não	Sim	S 40 a 49 teclas multifuncionais	Fita cassete	Impressora, joystick
Apple	TK 2000 II, Maxxi, Apple II Plus, Micro-Engenho, Craft II Plus, Elppa II Plus, Exato-Pro, AP II, AP II Ti etc.	6502	DOS, CP/M	12K	48K	Até 64K (linear) e até 512K (chaveada)	Não	Sim	Sim	M 52 teclas multifuncionais em alguns modelos	Fita cassete ou disquete	Impressora, joystick, paddles, caneta óptica, tablete gráfico, sintetizador de som e de voz, mouse, modem
TRS-80	CP 300, CP 500 Sysdata III, Sysdata Jr., DGT 100, DGT 1000 etc.	Z80A	DOS, CP/M	14K	48K	64K (com CP/M)	Sim	Não	Sim	M 65 teclas	Fita cassete ou disquete	Impressora, modem
IBM PC	PC 2001, EGO, Nexus, Craft XT, Diginet XT, I 7000 PCXT etc.	8088	DOS, CP/M, Unix	40K	60K	Até 576K	Sim	Sim	Sim	C 83 teclas, com 10 de função	Fita cassete, disquete ou disco rígido	Impressora, joystick, paddles, caneta óptica, tablete gráfico, sintetizador de som e de voz, mouse, modem, disco rígido

Linha Itaú

O microcomputador Itaú I-7000 é um projeto inteiramente nacional. Não se filia a nenhuma das quatro famílias principais. Produzido em 1983 pela Itautec — Itaú Tecnologia S.A., constitui uma linha à parte, formada pelo I-7000 e o I-7000 PC XT — este compatível com o I-7000, o IBM PC e o IBM XT.

De ampla utilização, as máquinas dessa linha interessam a profissionais e empresas de pequeno e médio porte. Têm recursos para teleprocessamento e funcionam também como terminais de mainframes (computadores centrais de grande porte).

O I-7000 trabalha com microprocessador de 8 bits e sistema operacional SIM/M, compatível com o CP/M-80. Sua RAM expande-se de 64 para 128 Kbytes. A ROM básica, de 4 Kbytes, conta com cartuchos de 32 Kbytes opcionais, para expansão. Dispõe de muitos recursos para armazenamento de dados: fitas cassete, disquetes de 5 1/4 polegadas ou de 8 polegadas e discos rígidos.

O ramo paralelo
Os equipamentos da Itautec formam uma linha à parte. São aparelhos versáteis, com tecnologia própria de hardware e software. Nas fotos, o I-7000 e uma impressora matricial.

Linguagens		Software disponível								Recursos para teleprocessamento	Acesso a bancos de dados remotos
Residentes	Disponíveis em disco	Jogos	Educacionais	Pacotes financeiros	Processadores de texto	Bancos de dados	Planilhas	Geradores de gráficos	Integrados		
ASSEMBLER, BASIC	—	S	S	S	S	S	S	—	—	—	—
ASSEMBLER, BASIC	LOGO Com CP/M: COBOL, FORTRAN, PASCAL, APL LISP, FORTH	C	C	M	C	C	C	C	—	Rede de micros, conexão com computadores de grande porte	Cirandão Videotexto
ASSEMBLER, BASIC	Com CP/M: COBOL, FORTRAN, PASCAL	S	S	M	M	C	M	—	—	Rede de micros, conexão com computadores de grande porte	Cirandão Videotexto
ASSEMBLER, BASIC	LOGO, COBOL FORTRAN, PASCAL, APL, LISP, FORTH, C, MODULA 2	C	M	C	A	A	A	A	A	Rede de micros, conexão com computadores de grande porte, acoplamento de terminais não inteligentes	Cirandão Videotexto

S = Recursos simples M = Médios C = Complexos A = Avançados — Dados de maio de 1985

VISÃO PERIFÉRICA

CHRIS STEVENS

A aquisição de um microcomputador é apenas o primeiro passo para a montagem de um sistema completo. O equipamento básico pode se ampliar de modo bem funcional pelo acréscimo dos acessórios mais convenientes.

Até pouco tempo atrás, o equipamento periférico mais procurado pelos possuidores de micros era um módulo de expansão da RAM. Em virtude do alto custo inicial dos chips de memória, os fabricantes de micros mais populares optaram por equipá-los com o mínimo de memória, a fim de facilitar o acesso ao consumidor. O ZX-81 sem expansão, por exemplo, dispunha de apenas 700 bytes de memória para a execução dos programas do usuário. Já os equipamentos mais recentes costumam vir com 48 ou 64 Kbytes de RAM ou mais. Com esse aumento de capacidade, os módulos de expansão da RAM tornaram-se menos necessários.

A variedade de periféricos à disposição do usuário hoje é enorme — desde os modems, que permitem a interligação de micros a distância, até as interfaces para controle de veículos e robôs e os sintetizadores de voz, que ampliam as possibilidades recreativas e educacionais dos micros.

Apesar disso, a maior parte dos periféricos é especialmente fabricada para os microcomputadores de larga penetração. Na hora da compra de um micro esse fator é decisivo, pois os modelos mais vendidos oferecem maior número de opções de periféricos. Nesse aspecto, também estão em desvantagem os equipamentos recém-lançados, que não contam com uma linha abrangente de periféricos. Contudo, espera-se pela introdução de padrões capazes de modificar essa situação, a fim de que os mesmos periféricos sejam compatíveis com todos os computadores construídos segundo os novos padrões.

A questão da compatibilidade dos periféricos é fundamental e deve ser considerada com critério não apenas no que diz respeito ao equipamento disponível, como também ao que venha a ser adquirido. Alguns periféricos disponíveis para o Spectrum da Sinclair, por exemplo, não funcionam com a Interface One *(ver ao lado)* ligada ao microcomputador.

Existindo compatibilidade entre os equipamentos, a utilização de periféricos aumentará bastante a versatilidade de seu computador; e a escolha adequada dos periféricos permite a montagem de um sistema de computação adaptado tanto às necessidades presentes como às futuras.

Sistemas de armazenamento de dados

O dispositivo de armazenamento mais usado nos microcomputadores é o gravador cassete comum. Isso porque as unidades de disco, ou drives, mais rápidos e confiáveis custam mais caro. Outro problema dos drives é o da compatibilidade. O micro BBC foge a essa regra, pois é dos poucos que se conectam a vários modelos diferentes de drive — um deles, o Torch Disc Pack. A maioria dos micros utiliza os disquetes de 5 1/4 polegadas. Contudo, a tendência à miniaturização já se faz sentir também nessas unidades com o lançamento dos slim drives (unidades de disco finas, cuja espessura equivale à metade da de uma unidade de disco comum) e a popularização dos disquetes de 3 e 3 1/2 polegadas.

Para os equipamentos mais populares, a Sinclair resolveu o problema do armazenamento optando por um sistema sem unidade de disco. Em seu lugar, desenvolveu o sistema Interface One/Microdrive, que funciona com uma fita em anel capaz de armazenar por volta de 85 Kbytes de dados.

A operação da fita é controlada pelo computador, possibilitando a localização de qualquer item em cerca de dez segundos. Seu desempenho, portanto, fica entre o de um gravador cassete e o dos drives, mas o preço é bem menor que o da unidade de disco mais barata.

Um dispositivo semelhante ao sistema Interface One/Microdrive é o Wafadrive Rotronics. O suporte para armazenamento de dados também é uma fita em anel, mas no sistema estão incluídas as interfaces RS232 e Centronics, além de um programa de processamento de texto.

Na foto, o Wafadrive Rotronics; o drive Oric/Atmo; e o sistema Interface One/Microdrive, da Sinclair.

Dispositivos gráficos

A geração de imagens gráficas por meio de microcomputador tornou-se uma tarefa simples graças ao desenvolvimento de diversos dispositivos. Os mais baratos são as canetas ópticas, que permitem desenhar diretamente na tela do vídeo. Esse recurso se deve à ação de uma célula fotoelétrica que detecta a posição da ponta da caneta sobre a tela. A partir dessa caneta desenvolveu-se o fuzil óptico Stack, que, nos jogos eletrônicos, substitui o joystick.

Para evitar o desenho a mão livre sobre a tela do vídeo, criaram-se os tabletes digitalizadores, em cuja superfície plana o usuário pode traçar linhas. Uma caneta especial transfere para o computador qualquer movimento realizado na superfície do tablete. Outro recurso disponível é o traçador gráfico, que, por meio de resistores variáveis instalados num braço mecânico, detecta a posição da caneta situada na extremidade do braço do aparelho.

Nas fotos, o tablete digitalizador Grafpad, da British Micro; o traçador gráfico Robot Plotter; e a caneta e o fuzil óptico Stack.

Modems

Os sinais analógicos transmitidos pela linha telefônica são convertidos em digitais e vice-versa, por meio dos modems (modulador/demodulador). Graças ao desenvolvimento de modems baratos, a interligação de micros pela rede telefônica se tornou uma realidade. A maior parte dos modems foi produzida para os microcomputadores equipados com a interface RS232 padronizada. Com o software apropriado, esses micros podem acessar o Videotexto ou algum banco de dados similar e obter informações especialmente elaboradas para o usuário não profissional. Os modems servem ainda para a comunicação com outros usuários pelo "correio eletrônico". Também neste caso, no entanto, é crucial a questão de compatibilidade entre os equipamentos — a velocidade de transmissão dos dados, medida em bauds, varia para cada banco de dados; e, em geral, os modems usados para acessar o Videotexto, por exemplo, não permitem a comunicação com o correio eletrônico.

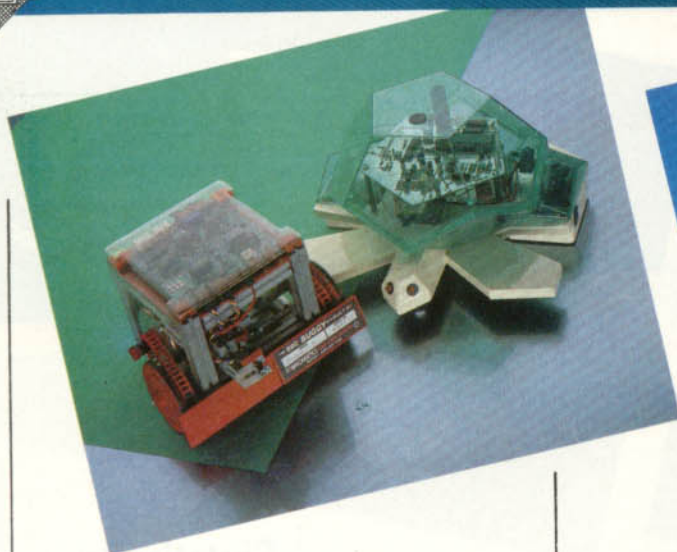
Nas fotos, o modem Prism VTX 5000 e o modem da Commodore.

Sintetizadores de voz

Muitos microcomputadores são capazes de gerar palavras acusticamente, com a simples conexão de um sintetizador de voz. Existem no mercado internacional dois tipos de sintetizador de voz: os de vocabulário fixo, com cerca de cem palavras (gravadas por uma pessoa), e os que produzem as palavras por meio de um conjunto de fonemas distintos e pausas. O número de palavras neste caso é ilimitado.

As aplicações dos sintetizadores de voz vêm se ampliando dia a dia: além dos jogos, já há automóveis cujo painel emite uma mensagem falada avisando que a gasolina está no fim ou que a porta não está bem fechada.

Os sintetizadores de voz também prestam valioso auxílio aos deficientes visuais.

Na foto, o Speech 64, da Currah; e o Sweetalker, desenvolvido pela Cheetah para o Spectrum.

Controladores programáveis

O controle de dispositivos e máquinas por meio de computadores não surpreende ninguém. Uma das aplicações mais citadas é a do controle dos aquecedores domésticos — mas, embora isso seja perfeitamente realizável, qual é a vantagem de se substituir os eficientes (e muito mais baratos) termostatos que já controlam esses sistemas? Muito mais divertido são os diversos dispositivos motorizados controlados por microcomputadores, a exemplo da Valiant Turtle. Capaz de traçar as imagens gráficas usadas na linguagem LOGO, a "tartaruga" é controlada pelo micro com raios infravermelhos. Um dispositivo semelhante, mas conectado ao computador por fios, é o Buggy, produzido pela BBC. Equipado com sensores, também serve para traçar linhas e gráficos.

Na foto, a Valiant Turtle e o Buggy, da BBC.

Impressoras/traçadoras

Nas impressoras matriciais, os caracteres são formados por um conjunto de pequenos pontos, feitos por agulhas que avançam ou se retraem. Esse sistema também serve para traçar gráficos. Embora as impressoras matriciais sejam bem mais rápidas, a qualidade da impressão é inferior à obtida com as impressoras tipo margarida. Estas últimas são, basicamente, máquinas de escrever controladas pelo computador, com os caracteres individuais colocados nas extremidades de uma roseta.

Um sistema opcional é a impressora/traçadora desenvolvida para os computadores Tandy, Atari, Commodore e Oric. Utilizando um rolo de papel com mais de 4 polegadas de largura, ela está equipada com quatro pequenas canetas esferográficas coloridas que desenham caracteres e gráficos. Processo diferente é empregado na Epson P40, impressora térmica que, com elementos aquecidos, "queima" os caracteres em papel especial.

Na foto, a impressora/traçadora na versão da Tandy Radio Shack e a impressora térmica Epson P40.

Joysticks

Nos jogos eletrônicos, a ação rápida dos participantes depende, em grande parte, dos dispositivos de controle manual — os joysticks —, que simulam o movimento de veículos e os mecanismos de direção e disparo de projéteis. Acoplados ao micro, os joysticks substituem o teclado, cuja utilização seria muito lenta para jogos desse tipo.

O largo emprego dos microcomputadores como opção de lazer, revelado pela grande procura de joysticks, forçou os fabricantes a acelerar os aperfeiçoamentos tecnológicos nesses periféricos. Graças a tal demanda (o joystick é o primeiro acessório comprado pela maioria dos usuários), muitos microcomputadores já saem da fábrica com as interfaces apropriadas; e alguns dos equipamentos mais recentes vêm também com os joysticks.

Os modelos mais comuns desse periférico têm conectores "D" com nove pinos, adotados inicialmente pelos micros Atari e Commodore e, depois, por inúmeras outras companhias. Os joysticks da BBC fogem a esse padrão, o que reduz bastante as opções de seus possuidores.

A Sinclair lançou no mercado, para seu micro Spectrum, a Interface Two, que funciona como interface para joysticks e adaptador para cartucho ROM do Spectrum. Outra possibilidade é a aquisição de uma interface "programável" que permite ao usuário rodar qualquer software, seja ou não originalmente desenvolvido para o uso de joysticks. O joystick, nesse caso, é usado para simular a ação do teclado. Esta é a melhor opção para o usuário do Spectrum, micro que apresenta grande coleção de jogos.

Na foto (da esquerda para a direita), o joystick da Amstrad; o RAT da Cheetah; o Kempston PRO 5000; e, na frente, a interface Kempston para o Spectrum ZX.

Monitores de vídeo

O monitor de vídeo dos microcomputadores costuma ser, pelo menos no início, o aparelho de televisão comum ligado a um conversor de radiofreqüência. Mas essa solução não é satisfatória se houver apenas um aparelho na casa. Além disso, a qualidade da imagem produzida pelo televisor nem sempre é das melhores. Assim, a solução é adquirir um monitor de vídeo, tomando muito cuidado para não escolher o aparelho errado, pois os monitores são fabricados de acordo com dois padrões — o RGB e o vídeo composto. Os monitores de vídeo do tipo composto são compatíveis com os micros das linhas Apple, TRS e IBM PC. Os computadores IBM PC podem usar monitores do tipo RGB, que produzem imagens de melhor qualidade. Alguns micros utilizam o alto-falante do televisor para a geração de sons e, portanto, necessitam de monitores equipados com alto-falantes. Na foto, o monitor de vídeo Microvitec Cub.

A ESCRITA NA TELA

Texto impecável
Depois de lidos e corrigidos na tela, os textos são transcritos em papel por uma impressora acoplada ao micro. As modificações introduzidas na cópia impressa são executadas na tela, redigitando-se apenas as correções.

Por meio de programas que manipulam textos, muitas pessoas têm seu primeiro contato com os micros. Conheça aqui as mais importantes aplicações e os recursos oferecidos por esse tipo de software.

Imagine o que significa redigir um manual técnico, uma apostila, uma tese ou mesmo um artigo de jornal. É necessário fazer muitas correções para que o texto fique claro, correto, bem organizado e adequado ao espaço que deve ocupar. Palavras e frases inteiras são modificadas ou eliminadas e novos conceitos, inseridos. Talvez seja preciso trocar a ordem de sentenças, parágrafos e até páginas inteiras.

Como fazer tudo isso dispondo apenas de papel e lápis ou de uma simples máquina de escrever? Os roteiristas de cinema, que com freqüência alteram a ordem das tomadas, costumam lançar mão de tesoura e cola. Outros profissionais recorrem a datilógrafas para refazer os textos, num trabalho repetitivo, que conta com poucos recursos para ficar correto e com boa estética.

A difusão dos microcomputadores trouxe o auxílio dos processadores (ou editores) de texto na resolução desses problemas. São programas que manipulam desde uma única letra até muitas páginas escritas, facilitando o trabalho de redação e correção de textos comerciais, técnicos, jornalísticos, didáticos e literários, além de correspondência e anotações pessoais.

Na utilização do microcomputador, os programas combinam os recursos de máquina de escrever, tela de vídeo e meios magnéticos de armazenamento de dados. Os textos são redigidos e alterados com facilidade na tela. Mediante combinações de teclas, que variam conforme o software, emitem-se comandos para inserir, eliminar, alterar, mover ou copiar letras, palavras, sentenças, parágrafos ou páginas inteiras.

Pronto, o texto e armazenado. Em geral o usuário grava-o em disquetes, que ocupam mui-

to menos volume que um arquivo de textos impressos. Os textos armazenados são lidos e corrigidos em tela sempre que necessário e, após a correção, transcritos em papel por uma impressora acoplada ao micro. Novas modificações, anotadas na cópia impressa, são em seguida executadas na tela. Altera-se apenas o que está impróprio; isso praticamente dispensa a redigitação. A correção fica facilitada por não se trabalhar com o texto no papel, mas em meios magnéticos. Obtém-se, assim, uma cópia final correta e com aparência impecável.

Entre os recursos mais comuns dos softwares atuais contam-se: alinhamento automático do texto pela direita; margens e tabuladores ajustáveis; mostrador do número de página, de linha e de coluna; marca automática de fim de página; centralização de títulos; negrito, sublinhado, subscrito (como no 2 de "H_2O") e superescrito (como em "1.º"), indicados por símbolos especiais.

Agora com completa **acentuação na tela**!
para brasileiro, alemão, ou sueco verem...
E francês também, bien sûr.
Não é preciso ser um expert em computação para usá-lo
Faça exatamente como se estivesse datilografando...
O que está em **negrito** na tela,
Aparece em **negrito** no papel.
Além disso, a versão nacional do WordStar oferece
documentação completa e todas telas em português fluente,
recursos de separação silábica, dicionário eletrônico,
mala direta e emissão de etiquetas com impressão condicional,
índices automáticos, edição com impressão simultânea...
e tem suporte para **todas as impressoras** nacionais e importadas.

Recursos amplos
O WordStar, campeão de vendas americano, foi adaptado para o Brasil com a introdução de caracteres da grafia portuguesa. Esse programa mostra o texto na tela como será impresso no papel. Os recursos dos processadores de texto são amplos: alguns programas dispõem até de buffer para armazenamento: enquanto a impressora produz um texto, o micro está liberado para processar outro (abaixo).

Nos processadores mais modernos, o texto aparece na tela exatamente como ficará na página impressa; o cabeçalho e o rodapé são automáticos (uma vez definidos, imprimem-se em todas as páginas). Recursos para encontrar e substituir determinadas palavras ou seqüências de caracteres permitem trocar uma palavra por outra no texto inteiro, com um único comando.

Por serem bastante elaborados, os softwares processadores de texto ocupam muito lugar na memória. Além disso, para obter resultados significativos, é necessário que haja acesso direto (aleatório) aos textos —, ou seja, a fim de visualizar certa informação na tela, basta indicá-la no diretório ou índice do disquete; o sistema operacional se encarrega de localizá-la.

Por isso, os microcomputadores que têm pouca capacidade de memória e utilizam fitas cassete dispõem de processadores de texto com recursos limitados, pois a fita só permite o acesso seqüencial: para localizar uma informação é necessário passar por todas as outras que foram gravadas antes dela, orientando-se pelo contador numérico do gravador.

No entanto, há progressos na área dos portáteis. Dotados de visor de cristal líquido, que mostra algumas linhas de cada vez, tais equipamentos podem armazenar os textos em fita, cartucho ou microfita, sendo ideais para jornalistas e vendedores. Esses profissionais utilizam o micro como um bloco de anotações e transmitem o texto via modem e linhas telefônicas para sua sede, distante até centenas de quilômetros.

Outro passo importante na simplificação do processo é o uso de um editor de texto previamente gravado num circuito integrado, que fica, portanto, residente no micro.

Os principais

Ao comparar diversos softwares para processamento de texto, devem-se verificar suas características quanto a facilidade de uso, formatação do texto na tela, impressão e manipulação de arquivos. Alguns programas têm mais recursos mas também ocupam mais memória, custam mais caro e apresentam muitos comandos.

Entre os processadores de texto de maior uso no Brasil está, para a linha TRS-80, o Electric Pencil, um dos primeiros no gênero. A linha Apple oferece, entre outros, os modelos Apple Writer, Magic Window e WordStar, este um campeão de vendas americano.

Há ainda o Editex, desenvolvido no Brasil especialmente para o MicroEngenho, da Spectrum. Esse programa permite verificar a ortografia e a acentuação. Enquanto o usuário escreve, o programa aplica as 132 regras que tem armazenadas. Se detectar uma infração a essas regras, fará soar um alarma e mostrará a forma correta. Também desenvolvido no Brasil é o Redator, destinado aos equipamentos Itautec.

Os programas nacionais têm recursos para produzir as letras e os sinais usados em português, como a cedilha, o til e os acentos; contudo, o hardware também deve estar capacitado para mostrar esses sinais na tela. Para isso dispõe de recursos como a interface Ivanita.

O equipamento mais bem servido de software para processamento de texto é o IBM PC e os compatíveis, pois seu sistema de processamento de 16 bits e sua grande capacidade de memória permitem a utilização de softwares poderosos e sofisticados, como o WordStar, o Word, o Multimate e o Symphony.

Este último faz parte de uma nova tendência na área, a dos softwares integrados, que reúnem os recursos de planilha eletrônica, gráficos, processador de textos, banco de dados e, em alguns casos, telecomunicações. Assim, com a tela dividida em "janelas", pode-se fazer com que uma carta incorpore dados de um gráfico, de um banco de dados ou de uma planilha financeira.

Alguns desses programas são direcionados por menus, isto é, a tela mostra permanentemente uma lista de funções mais utilizadas, indicadas por palavras ou, em alguns casos, por figuras.

Outros oferecem comandos fáceis de memorizar, com teclas associadas à letra inicial da palavra de comando; outros, ainda, têm os comandos principais associados às teclas de função, que são teclas especiais encontráveis nos computadores de maiores recursos, programáveis para substituir toda uma série de comandos. Convém que o programa mostre mensagens de erro esclarecedoras ou disponha de telas explicativas de auxílio ao usuário.

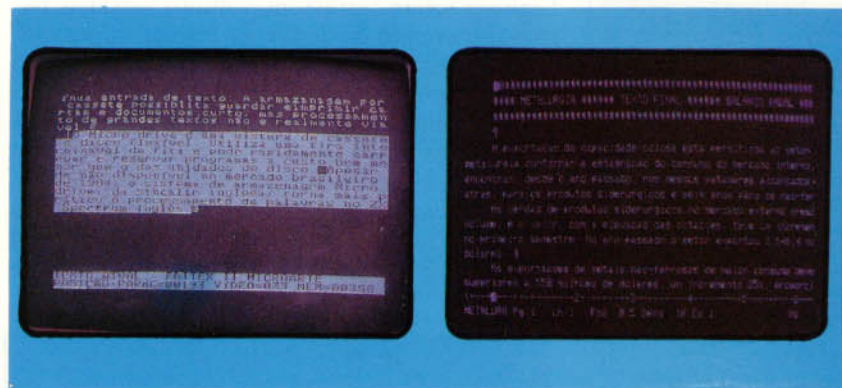
Criação e organização

Quem lida com aplicações comerciais conta com recursos específicos. Eles permitem, por exemplo, elaborar textos complexos, repetitivos e que exigem precisão, como contratos, fórmulas jurídicas e descrição de produtos. No caso, esses textos são compostos pela junção de vários parágrafos padronizados.

Pode-se ainda enviar a mesma carta a grande número de pessoas, mudando apenas as informações referentes ao destinatário. Tais cartas são produzidas com os softwares de Mala Direta (Mailmerge). Oferecidos por diversos processadores, esses recursos permitem imprimir muitas cópias da mesma carta, cada qual associada a um dos registros de arquivo de nomes e endereços. Da mesma forma, é possível redigir uma carta padrão na qual variam apenas certos dados.

Os programas mais sofisticados já possibilitam escrever cartas condicionais, nas quais alguns segmentos do texto dependem de variáveis encontradas no arquivo de endereços. Assim, por exemplo, conforme o sexo do destinatário, a carta dirá "Prezado Senhor" ou "Prezada Senhora", e uma carta de cobrança pode adaptar o prazo limite e o valor do pagamento aos dados referentes ao cliente.

Há programas que manipulam arquivos com grande rapidez; outros gravam continuamente o texto no disquete, evitando que se percam informações no caso de faltar energia elétrica ou falhar o equipamento.

Alguns programas dispõem de um buffer (reservatório) para armazenamento, com o que libertam o micro para processamento de um texto enquanto outro texto é transmitido do buffer para a impressora. Há também programas que enfileiram "ordens de serviço" para que vários textos sejam impressos em seqüência.

Ferramenta poderosa
O Editex, desenvolvido no Brasil, permite verificar a ortografia e a acentuação. O programa dispõe de 132 regras armazenadas.
Auxiliares eficientes em diversas atividades, os editores de texto ajudam agora o trabalho jornalístico, possibilitando produzir e enviar à redação em poucos minutos um texto final como o da ilustração à direita.

Os equipamentos de 16 bits já dispõem de novos tipos de software, à venda no mercado internacional, que vão além dos aspectos mecânicos da escrita e revisão dos textos, oferecendo recursos de auxílio à própria criação e organização das idéias. O THOR (THought ORganizer), armazena informações relativas a várias áreas ou projetos diferentes, classifica-as e indexa-as de acordo com diferentes critérios. Depois são pesquisadas, agrupadas e acessadas as informações selecionadas, na tela ou na impressora. Sem ser propriamente um banco de dados, o THOR oferece excelentes recursos de armazenamento e referência cruzada de textos.

Outra poderosa ferramenta é o WANDAH (Writing Aid AND Author's Helper), que auxilia a elaboração de um texto desde o planejamento inicial e a divisão do assunto em itens e subitens até a revisão final de ortografia, pontuação, concordância e mesmo estilo. Adverte o autor, por exemplo, quanto à repetição excessiva ou ao uso indevido de termos.

Com isso, chega-se ao limiar da era da inteligência artificial aplicada à linguagem escrita. Reunindo um grande corpo de conhecimentos dos processos complexos e muitas vezes irracionais da linguagem humana, os processadores de texto serão auxiliares cada vez mais eficientes de todos aqueles que trabalham com palavras.

ESCOLHA DE ARQUIVOS

O baixo preço de cartuchos e fitas cassete deu-lhes a preferência dos programadores quando se trata de armazenar dados. Mas os modernos discos flexíveis já se tornaram acessíveis a todo tipo de usuário.

CONTATO PERIGOSO

Nunca deixe o disquete próximo a imãs. Dispositivos como os telefones possuem eletroimãs e os alto-falantes dos sistemas de som são dotados de potentes imãs, capazes de alterar os dados gravados.

Embora sejam ferramentas muito versáteis para o tratamento de dados, os microcomputadores de nada servem se não dispuserem de um meio de reter a informação. Nos sistemas de armazenamento de dados incluem-se os cartuchos de ROM, as fitas cassete comuns e os discos magnéticos flexíveis, ou disquetes.

Preço, capacidade de armazenamento e facilidade de acesso aos dados são fatores que determinam a opção por um ou outro tipo de memória.

Unidade de disquete BBC
Para acoplá-la ao micro BBC é preciso instalar o DOS ROM no próprio computador. Esse procedimento é desnecessário nas unidades de disco "inteligentes", as quais já vêm equipadas com um circuito integrado DOS.

MEMÓRIAS DE CURTO E DE LONGO PRAZO
A memória de acesso aleatório (RAM, Random Access Memory) armazena as informações que podem ser alteradas pelo usuário. É, portanto, uma memória de curto prazo. A memória somente de leitura (ROM, Read Only Memory) contém as informações permanentes, que não podem ser alteradas. Trata-se de uma memória de longo prazo.

Cartucho

O armazenamento em cartucho apresenta poucas características favoráveis ao programador. Na maior parte dos cartuchos, a memória é programável apenas para leitura (PROM, Programmable Read Only Memory). Esse tipo funciona somente como meio de introdução de dados no computador, em geral sob a forma de jogos, escritos no extenso e complexo código de máquina. A PROM funciona também como recurso suplementar — por exemplo, extensões para o BASIC. Mas já foram desenvolvidos cartuchos com PROM apagáveis eletricamente (EEPROM, Electrically Erasable Programmable Read Only Memory), que podem ser gravadas e lidas do mesmo modo que a RAM interna. Ao contrário desta, contudo, a EEPROM não é volátil: a informação permanece registrada quando se removem os cartuchos ou quando se desliga o computador.

Há outros cartuchos de características semelhantes, disponíveis apenas para alguns computadores. Operando com circuitos de RAM que utilizam semicondutor complementar de óxido metálico (CMOS, Complementary Metal Oxide Semiconductor), tais cartuchos consomem pouca energia e conseguem manter a informação armazenada graças a uma bateria instalada em seu interior.

O principal argumento contra o armazenamento de dados em EEPROM e RAM CMOS é o alto preço desses tipos de memória. A formação de uma biblioteca de tais cartuchos, mesmo modesta, é quase tão dispendiosa quanto a aquisição de uma unidade de discos flexíveis.

Fitas cassete

Baratas e facilmente disponíveis, as fitas cassete funcionam nos gravadores portáteis, que podem ser conectados a quase todos os microcomputadores. Por isso, continuam a ser o meio mais empregado para o armazenamento de programas e dados.

O armazenamento se faz em arquivos seqüenciais, sob a forma binária: os zeros e os uns correspondem a tons diferentes. A gravação do nome do arquivo, de seu endereço na memória e de seu conteúdo é realizada bit a bit em grupos de 1 byte (8 bits) e segmentos de 256 bytes. Alguns computadores contam com um recurso denominado "soma de verificação", que permite verificar eventuais erros de gravação.

Os comandos básicos em BASIC nesse sistema de armazenamento, são SAVE, para gravar os arquivos na fita, e LOAD, que permite o acesso aos dados e a leitura deles.

Em alguns sistemas há comandos adicionais para várias funções específicas, como gerar um catálogo com os nomes dos arquivos armazenados ou formatar a gravação e leitura de dados de diferentes tipos.

O baixo custo e a facilidade de compreensão dos comandos são vantagens pequenas quando comparadas aos vários inconvenientes desse tipo de armazenamento.

Na grande maioria dos casos, o usuário se vê obrigado a operar manualmente os controles para o acesso aos dados. É necessário, assim, um ajuste preciso do volume do gravador e muito cuidado no acionamento das teclas, o que torna a operação um tanto demorada.

Outro inconveniente diz respeito ao modo seqüencial do armazenamento de dados em fitas.

Se quiser recuperar um arquivo específico, o operador deverá estar atento ao contador de giros do gravador para voltar a fita na posição exata (um pouco antes do início do arquivo). Do contrário, o computador será obrigado a pesquisar toda a fita. E dentro de cada arquivo é impossível recuperar os itens de imediato, sem precisar processá-lo por inteiro.

A todas as desvantagens relacionadas, soma-se a baixa velocidade de gravação e leitura do sistema: de 300 a 1.200 bits por segundo. Um programa curto, de 5 Kbytes, por exemplo, pode levar de 1 a 3 minutos para ser carregado ou gravado. A baixa velocidade do sistema dificulta também a obtenção das valiosas cópias de reserva (backup) dos programas.

Desgaste da fita, provocando imprevisíveis alterações nos dados, e possibilidade de rupturas devidas ao sistema de transporte da fita completam a lista das desvantagens desse tipo de armazenamento.

Disco flexível

Comparados aos sistemas de armazenamento em cartucho ou cassete, os discos (ou disquetes) flexíveis apresentam poucos inconvenientes. As unidades de disquete (drives) são aparelhos complexos e delicados — o que torna seu preço consideravelmente elevado. Os disquetes também são caros, mas para o usuário constituem um meio confiável e versátil de armazenar grande quantidade de dados. A velocidade de gravação e leitura em discos flexíveis chega a ser até duzentas vezes maior do que a das fitas cassete.

Todas as unidades dispõem de algum tipo de sistema operacional de disco (DOS, Disk Operating System), que contém uma rotina responsável pela distribuição dos dados pelas trilhas do disco. Em geral, existem de 35 a 80 trilhas por lado, cada uma dividida em número variável de setores. As trilhas próximas ao centro do disco são menores e possuem menos setores do que as externas. Cada setor, por sua vez, abrange um bloco de dados, em geral de 256 bytes.

O DOS registra as posições em que estão gravados todos os dados armazenados no disco. Isso é possível graças à existência de um mapa de disponibilidade de blocos (BAM, Block Availability Map) — gravado no próprio disco ou na memória principal do computador — e de um catálogo (ou diretório). O BAM mantém um registro dos blocos em uso e dos que estão livres para novo armazenamento; o catálogo é uma lista dos nomes e tipos de arquivo e da localização de cada um por trilha e por setor. Costuma-se gravá-lo na trilha central e também carregá-lo na memória do computador, para funcionar como referência. Após consultar o BAM, o DOS direciona o movimento da cabeça de leitura/gravação para que se efetuem as operações de armazenamento e recuperação de dados.

A alocação dos dados em trilhas e setores e o posicionamento exato da cabeça de leitura/gravação tornam possível o acesso direto aos arquivos. Se necessário, os bits podem ser gravados ou lidos em grupos de até 1 byte. De modo geral, as diferenças entre as unidades de disquete são pequenas: limitam-se à quantidade de dados — de 100 a 400 Kbytes — que pode ser armazenada; à velocidade de transferência dos dados; e às maneiras de se controlar o registro e a leitura dos dados por meio do DOS.

Há três métodos principais para se instalar um DOS. O mais eficiente é implantá-lo, em forma de ROM, na unidade de disco, sob o controle do microprocessador da própria unidade, associado à RAM. Essas unidades de disco são chamadas "inteligentes", pois, ao receberem uma instrução da unidade central de processamento (CPU), podem executar independentemente as complexas rotinas de operação dos discos. Assim, o processador do microcomputador fica liberado para continuar executando o programa principal.

O método mais comum, no entanto, é o que transfere o DOS, armazenado no disco, para a RAM do computador. Isso pode ser feito automaticamente ou por meio de um comando manual, quando se aciona o computador.

O último método consiste na inclusão de um tipo de DOS no próprio sistema operacional do computador (gravado numa EPROM). As rotinas para a operação dos discos incluem os comandos do tipo SAVE e LOAD, um comando CAT (catalog) para o diretório, um comando para a formatação do disquete (ou do cartucho) e vários comandos para criação, manipulação e eliminação de arquivos seqüenciais ou de acesso direto.

Gravação em disquete
Os disquetes são feitos de substância plástica resistente, coberta por uma camada magnetizável de óxido metálico. Protegido por um envelope plástico quadrado, o disco é girado por meio de seu eixo. Uma abertura no envelope **(6)** permite o acesso da cabeça de leitura/gravação à superfície de registro.

1 ENVELOPE PROTETOR
2 ENTALHE DE PROTEÇÃO CONTRA GRAVAÇÃO
3 SETOR
4 ORIFÍCIO DE ALINHAMENTO
5 TRILHA
6 ABERTURA DE ACESSO À CABEÇA DE LEITURA/GRAVAÇÃO

LIÇÕES E DESAFIOS

Uma série de softwares educativos exemplificam o que a informática pode fazer nessa área e quais os erros a serem evitados pelo programador voltado para a criança.

Os jogos aqui examinados têm objetivos educacionais que variam desde o simples reconhecimento de cores e formas até o desenvolvimento da criatividade infantil, passando pelas habilidades básicas de leitura e manipulação de números. Todos eles explicam ao jovem usuário, de modo direto, o que fazer.

Esta deve ser uma prioridade em qualquer programa educativo: a criança precisa compreender o que lhe cabe fazer, recebendo recompensas bem definidas toda vez que domina determinada técnica.

Convém igualmente que o programa seja fácil de usar. Não haveria utilidade num software que se destinasse a ensinar a criança a ler e começasse com uma lista de instruções operacionais. Os melhores programas apresentam um mínimo desse tipo de instrução inútil e, não raro, complicadora.

Um bom programa educativo é o que mantém o interesse da criança. Por mais valioso que seja seu objetivo, não o atingirá se estiver acima da capacidade do usuário ou se for repetitivo.

Outra exigência fundamental do programa educativo: que ele de fato ensine o que se propõe ensinar. Isso parece evidente, mas muitas vezes os fabricantes de software esquecem o objetivo educacional ao favorecer o valor de entretenimento.

Os pacotes aqui apresentados, produzidos pelas companhias americanas Spinnaker e Fisher-Price, são encontrados em cartuchos ou fitas cassete no mercado internacional. A criança pequena aprende com facilidade a colocar um cartucho no computador e ligá-lo. Já a fita cassete exige a presença de um adulto para ajudar a carregar o programa.

Fantasia Dançante

Danças e viagens

Dance Fantasy (Fantasia Dançante) tem como público-alvo crianças de quatro a oito anos. A imagem na tela mostra um palco no qual estão duas figuras em pé. A criança é convidada a coreografar uma dança para elas. Antes de começar, ela especifica o sexo dos dançarinos: um menino e uma menina, dois meninos, duas meninas.

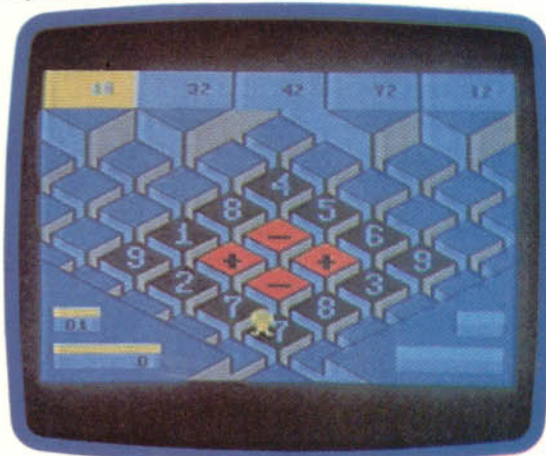
Na parte inferior da tela aparecem figuras em várias poses: cada uma representa determinada seqüência de dança (salto, pirueta etc.). Ao mover cada dançarino sobre uma dessas figuras, usando o joystick, a criança escolhe a seqüência específica. Depois posiciona as figuras no palco e, pressionando o botão de disparo, faz com que elas dancem.

Viagem pelo Egeu

Assim, a dança é coreografada pela seleção de uma série de movimentos e de sua execução em pontos diferentes do palco. O programa fornece os movimentos de ligação. Completada a dança, a criança pode gravá-la e visualizar o efeito final.

A força de Dance Fantasy reside em sua imaginosa analogia com um programa de computador: a criança pode montar sua própria dança *(programa)* usando uma série de seqüências básicas (um conjunto de *rotinas)*. Depois *grava* o resultado em fita cassete e o *carrega* — assim, sem dificuldade, se familiariza com esses conceitos.

Aegean Voyage (Viagem pelo Egeu) é para crianças um pouco maiores. Usa personagens e cenários da mitologia grega como elementos de um jogo de aventuras. O objetivo: fazer um navio viajar de Atenas até várias ilhas do mar Egeu, evitando recifes e tempestades. Quando o jogador chega à segurança de um porto, surgem na tela o nome da ilha e uma mensagem cifrada. Ele então decide se a ilha deve ser explorada. Se a decisão é correta, recebe como recompensa um tesouro (por exemplo, o escudo de Aquiles); se é errada, recebe punição: o afundamento do navio por uma criatura como a Górgona.

O estudioso dos clássicos notará que o jogo apresenta alguns fatos errados: o Minotauro, por exemplo, aparece tanto em Delos como em Creta. Além disso, o jogo não explica o significado de nomes e lugares. É mínima a probabilidade de a criança obter conhecimentos razoáveis da mitologia grega com esse cassete.

Jogando com Números

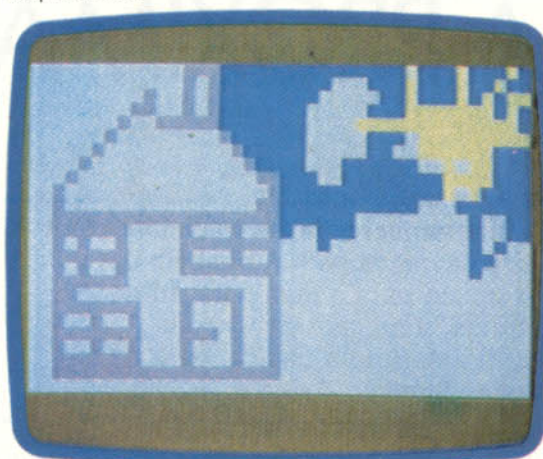
Ele não mantém a atenção da criança por muito tempo, pois as formas e os gráficos são repetitivos e pouco interessantes.

Números, nomes e formas

Projetado para desenvolver a habilidade aritmética de garotos de oito a doze anos, Number Tumblers (Jogando com Números) proporciona a velocidade e a emoção de um fliperama. Apresenta na parte de cima da tela uma série de números, e o jogador deve ordenar os símbolos aritméticos e numéricos nos lados de um conjunto de dados, para criar uma expressão matemática que corresponda a um dos números.

É um jogo rápido e divertido, com gráficos atraentes e bem desenhados. Um incentivo real para a criança desenvolver a habilidade em aritmética.

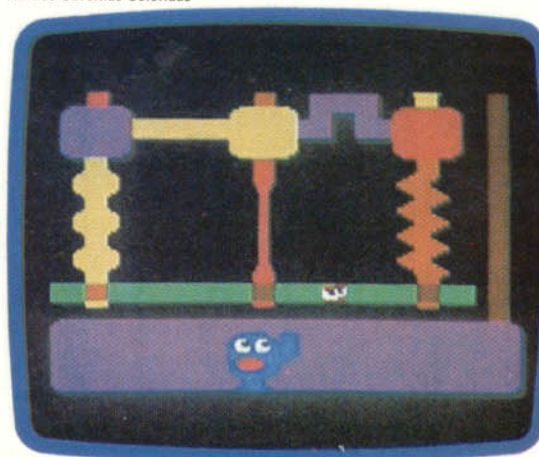
Kindercomp (Computador Infantil) destina-se a meninos de três a oito anos. O pacote — com uma série de diferentes exercícios — visa a familiarizar a criança com o uso do computador e a desenvolver suas aptidões artísticas. Mas dá a impressão de se limitar a uma série de truques com o computador — o tipo de coisa que a maioria dos programadores faz ao aprender BASIC e testar as capacidades da máquina.

Computador Infantil

Numa das opções, Nomes, o programa convida o jogador a fornecer um nome ou uma sentença de até quinze caracteres de comprimento. As letras aparecem em cores e tamanhos variados por toda a tela. O efeito visual, para a criança não acostumada com os gráficos do computador, parece fabuloso. Contudo, o programa tem pouco valor educativo, pois qualquer grupo de letras proporciona o mesmo efeito.

O Kindercomp pode não manter o interesse da criança por muito tempo — seus truques são divertidos mas limitados. É o tipo de pacote com que os garotos se entretêm por uns três dias — e depois nunca mais usam.

Alf In The Color Caves (Alf nas Cavernas Coloridas) apresenta um divertido personagem — Alf — que escorrega por uma série de tubos coloridos e de várias formas até cair numa sala no fundo das cavernas. Usando um joystick ou o teclado, o jogador guia o personagem pelas cavernas. Se o conduzir com segurança, passando por ameaçadores pares de olhos que se movem rapidamente, será recompensado com o espetáculo de uma graciosa dança de Alf. Depois, o personagem é sugado por um tubo até o chão, para reiniciar a descida.

Esse é um programa educativo funcional, adequado a crianças pequenas. Pode servir de exemplo aos produtores dos softwares destinados ao desenvolvimento de habilidades infantis.

Alf nas Cavernas Coloridas

A PRÓXIMA LETRA

A programação desse tradicional jogo de palavras constitui também um bom exercício para a manipulação de strings. Aqui, as três versões de sua estrutura básica: para as linhas Sinclair, Apple e TRS-80.

Poucos jogos são tão conhecidos em todo o mundo quanto o de forca. Na versão eletrônica desse passatempo, o jogador tem por objetivo adivinhar, letra por letra, uma palavra escolhida aleatoriamente pelo micro entre onze determinadas pelo programador. Cada palavra tem no máximo dez letras, e os espaços correspondentes estão indicados por traços horizontais na tela. Todas as letras tentadas pelo jogador permanecem na tela, lembrando que já foram usadas. Quando o jogador acerta uma letra, ela,é colocada no lugar a que pertence na palavra. A cada tentativa errada, o micro desenha uma das dez partes que formam a figura de um homem. Se o jogador cometer dez erros, o boneco é enforcado: o jogador perde a partida e o aparelho seleciona uma nova palavra.

Na versão para a linha Sinclair, as palavras não estão armazenadas no programa; elas são digitadas pelo programador, no início de cada jogo. Para iniciar, digite RUN [NEW LINE] e uma mensagem solicitará onze palavras para que uma seja sorteada.

É fácil, nas três versões, modificar o número de letras por palavra, assim como o número total de palavras.

Como um desafio extra, você pode melhorar seu programa, acrescentando, por exemplo, uma melodia de congratulações ou de pêsames; ou tentar fazer o enforcado balançar, ao fim do jogo.

Um momento do jogo
Na tela, a evolução do programa, mostrando as tentativas já feitas (A, C, D, H, O), a letra correta (O) entre os traços e as partes do corpo desenhadas, além do aviso do computador de que já foi tentada a letra C.

TRS-80

```
10  REM ******************************
11  REM *      JOGO DA FORCA         *
12  REM ******************************
20  CLEAR 1000:DIM P$(12)
30  REM ***  INICIALIZAR ***
40  D$="----------"
50  FOR I=1 TU 11
60  READ P$(I)
70  NEXT I
80  REM ***  NOVA PALAVRA  ***
90  CLS
100 PRINT @30,"FORCA"
110 P$=P$(RND(11))
120 C=0:P=0:D=0
130 G$=""
140 PRINT @768,"LETRAS: ";
150 L=LEN(P$)
160 PRINT @640,LEFT$(D$,L)
170 REM ***  INPUT LETRA  ***
180 PRINT @896,"ESCOLHA A LETRA: ";STRING$(45,32):PRINT
    @913,"";:INPUT L$
190 REM ***  VERIFICA LETRA  ***
200 IF L$="" THEN GOTO 180
210 IF LEN(L$)>1 THEN GOTO 180
220 F=0
230 FOR I=1 TO LEN(G$)
240 IF L$<>MID$(G$,I,1) THEN GOTO 270
250 PRINT @913,"VOCE JA USOU ESTA LETRA";:FOR X=1 TO
    1000:NEXT X
260 F=1
270 NEXT I
280 IF F=1 THEN GOTO 170
290 G$=G$+L$
300 PRINT @776,G$;
310 REM *** TESTE DA LETRA NA PALAVRA ***
320 F=0
330 FOR I=1 TO L
340 IF L$=MID$(P$,I,1) THEN GOSUB 500
350 NEXT I
360 IF F<>1 THEN GOSUB 800
370 IF D=1 THEN GOSUB 700:GOTO 80
380 IF C=L THEN GOSUB 600:GOTO 80
390 GOTO 170
400 END
500 REM *** ACERTOU A LETRA ***
510 PRINT @639+I,L$;
520 C=C+1
530 F=1
540 RETURN
600 REM *** ACERTOU A PALAVRA ***
610 PRINT @896,"PARABENS!!!        ACERTOU A PALAVRA";
    STRING$(27,32);
620 FOR I=1 TO 3000:NEXT I
630 RETURN
700 REM *** PERDEU O JOGO ***
710 PRINT @896,"ENFORCOU-SE! A PALAVRA ERA  ";P$;
720 FOR I = 1 TO 2000:NEXT I
730 RETURN
800 REM *** ERROU A LETRA ***
810 P=P+1
820 ON P GOTO 830,840,850,870,880,890,900,910,920,930,
    940
830 REM *** DESENHA CABECA ***
831 PRINT @50,CHR$(151);CHR$(131);CHR$(131);CHR$(131);
    CHR$(171);:PRINT@114,CHR$(149);"   ";CHR$(170);:
    PRINT @178,CHR$(141);CHR$(140);CHR$(140);CHR$(140);
    CHR$(142);:RETURN
840 REM *** DESENHA PESCOCO ***
841 PRINT @180,CHR$(188);:PRINT @244,CHR$(131);:RETURN
850 REM *** DESENHA CORPO ***
851 PRINT @241,CHR$(151);CHR$(131);CHR$(131);CHR$(131);
    CHR$(131);CHR$(131);CHR$(171);:PRINT@305,CHR$(149);
    STRING$(5,32);CHR$(170);:PRINT @369,CHR$(149);
    STRING$(5,32);CHR$(170);
860 PRINT @433,CHR$(149);STRING$(5,32);CHR$(170);:
    PRINT @497,CHR$(131);CHR$(131);CHR$(131);
    CHR$(131);CHR$(131);CHR$(131);CHR$(131);: RETURN
870 REM *** DESENHA BRACO ESQUERDO ***
871 PRINT @237,CHR$(131);CHR$(131);CHR$(131);CHR$(131);
    RETURN
880 REM *** DESENHA BRACO DIREITO ***
881 PRINT @248,CHR$(131);CHR$(131);CHR$(131);CHR$(131);
    RETURN
890 REM *** DESENHA PERNA ESQUERDA ***
891 PRINT @497,CHR$(171);:PRINT @561,CHR$(170);:PRINT
    @625,CHR$(170);:RETURN
900 REM *** DESENHA PERNA DIREITA ***
901 PRINT @503,CHR$(151);:PRINT @567,CHR$(149);:PRINT
    @631,CHR$(149);:RETURN
910 REM *** DESENHA PE' ESQUERDO ***
911 PRINT @624,CHR$(176);CHR$(186);:RETURN
920 REM *** DESENHA PE' DIREITO ***
921 PRINT @631,CHR$(181);CHR$(176);:RETURN
930 REM *** DESENHA MAO ESQUERDA ***
931 PRINT @237,CHR$(151);:RETURN
940 REM *** DESENHA MAO DIREITA ***
941 PRINT @251,CHR$(171);:D=1:RETURN
950 REM *** PALAVRAS SECRETAS ***
960 DATA "TIGELA","QUERUBIM","EXORCISTA","XENOFOBIA",
    "ARCEBISPO"
970 DATA  "ISTMO","ESTUPIDO","ENFORCADO","ESTRANHO",
    "REVISTA","MASCULO"
```

Sinclair

```
  5 REM ***********************
 10 REM *    JOGO DA FORCA     *
 15 REM ***********************
 20 REM ***  ENTRADA PARA 11 PALAVRAS  ***
 30 LET N=11
 40 DIM X$(N,10)
 45 DIM Y(N)
 50 LET D$="----------"
 55 PRINT "DIGITE A PALAVRA"
 60 FOR I=1 TO N
 65 PRINT AT 2,8;I
 70 INPUT W$
 75 LET Y(I)=LEN W$
 80 LET X$(I)=W$
 90 NEXT I
100 CLS
110 PRINT AT 1,1; "JOGO DA FORCA"
120 LET I=1+INT (RND*11)
130 LET W$=X$(I)
140 LET C=0
150 LET W=0
160 LET G$=""
170 LET D=0
180 PRINT AT 20,1; "USADAS : "
185 LET L=Y(I)
190 PRINT AT 16,1;D$(1 TO L)
200 REM ***  ENTRADA DA LETRA  ***
220 INPUT L$
230 IF L$ = " " THEN GOTO 220
240 IF LEN L$>1 THEN GOTO 220
250 LET F=0
270 FOR I=1 TO LEN G$
280 IF L$<>G$(I) THEN GOTO 330
290 PRINT AT 21,1; "ESTA VOCE JA TENTOU..."
310 PAUSE 100
315 PRINT AT 21,1; "                        "
320 LET F=1
330 NEXT I
340 IF F=1 THEN GOTO 220
350 LET G$=G$+L$
360 PRINT AT 20,10;G$
370 REM ***  TESTE DA LETRA NA PALAVRA  ***
380 LET F=0
390 FOR I=1 TO L
400 IF L$=W$(I) THEN GOSUB 480
410 NEXT I
420 IF F<>1 THEN GOSUB 670
430 IF D=1 THEN GOSUB 590
435 IF D=1 THEN GOTO 100
440 IF C=L THEN GOSUB 530
445 IF C=L GOTO 100
450 GOTO 220
460 STOP
470 REM ***  A LETRA COMBINA  ***
480 PRINT AT 16,I;L$
490 LET C=C+1
500 LET F=1
520 RETURN
530 REM ***  SUCESSO...  ***
540 PRINT AT 10,1;"***VOCE VENCEU***"
550 PAUSE 100
560 PRINT AT 10,1;"                  "
570 RETURN
580 REM ***  FRACASSO...  ***
590 PRINT AT 10,1; "***VOCE PERDEU***"
600 PAUSE 100
610 CLS
630 PRINT AT 10,3; "A PALAVRA ERA : ";W$
640 PAUSE 100
650 RETURN
660 REM ***  NAO E A LETRA  ***
670 LET W=W+1
680 IF W=1 THEN GOTO 700
681 IF W=2 THEN GOTO 750
682 IF W=3 THEN GOTO 800
683 IF W=4 THEN GOTO 850
684 IF W=5 THEN GOTO 900
685 IF W=6 THEN GOTO 950
686 IF W=7 THEN GOTO 1000
687 IF W=8 THEN GOTO 1050
688 IF W=9 THEN GOTO 1100
689 IF W=10 THEN GOTO 1150
700 REM ***  CORDA ***
720 PRINT AT 2,24; "*"
730 PRINT AT 3,24; "*"
745 RETURN
750 REM ***  CABECA  ***
755 PRINT AT 4,22; "*****"
760 PRINT AT 5,22; "*   *"
770 PRINT AT 6,22; "*   *"
780 PRINT AT 7,22; "*   *"
785 PRINT AT 8,22; "*****"
790 RETURN
800 REM ***  OLHOS E BOCA  ***
810 PRINT AT 5,23; "0 0"
820 PRINT AT 7,24; "-"
830 RETURN
850 REM ***  PESCOCO  ***
860 PRINT AT 9,24; "*"
870 PRINT AT 10,24; "*"
880 RETURN
900 REM ***  PARTE DO CORPO  ***
910 PRINT AT 11,22; "*****"
920 PRINT AT 12,22; "*   *"
930 PRINT AT 13,22; "*   *"
940 RETURN
950 REM ***  RESTO DO CORPO  ***
960 PRINT AT 14,22; "*   *"
970 PRINT AT 15,22; "*   *"
980 PRINT AT 16,22; "*****"
990 RETURN
1000 REM ***  BRACO  ***
1010 PRINT AT 11,19; "***"
1020 RETURN
1050 REM ***  OUTRO BRACO  ***
1060 PRINT AT 11,27; "***"
1070 RETURN
1100 REM ***  PERNA  ***
1110 PRINT AT 17,22; "*"
1120 PRINT AT 18,22; "*"
1130 PRINT AT 19,22 "*"
1140 RETURN
1150 REM ***  OUTRA PERNA  ***
1160 PRINT AT 17,26; "*"
1170 PRINT AT 18,26; "*"
1180 PRINT AT 19,26; "*"
1190 LET D=1
1200 RETURN
```

Apple

```
]LIST

100  REM   ******************
110  REM   *  JOGO DA FORCA  *
120  REM   ******************
125  REM
130  TEXT
135  CLEAR : DIM P$(20)
140  GOSUB 490
145  VTAB 21: HTAB 1: PRINT "TECLE UMA LETRA [ ]"
150  VTAB 21: HTAB 18: GET T$:N = 0:B$ = ""
155  LET SOM =  PEEK ( - 16322): FOR I = 1 TO 26
160  IF T$ =  MID$ (A$,I,1) THEN N = I:I = 26: GOTO 170
165  IF I = 26 THEN  GOSUB 595
170  NEXT
175  FOR I = 1 TO 26
180  IF I = N THEN B$ = B$ + " ": GOTO 190
185  LET B$ = B$ +  MID$ (A$,I,1)
190  NEXT
195  VTAB 23: HTAB 1:A$ = B$: PRINT A$
200  REM  *** EXISTE ESTA LETRA ? ***
205  LET C = 0: FOR I = 1 TO  LEN (P$(P))
210  IF T$ =  MID$ (P$(P),I,1) THEN  GOSUB 615
215  NEXT
220  LET TA = TA + C: IF TA =  LEN (P$(P)) THEN  GOTO 435
225  IF C =  > 1 THEN  GOTO 145
230  LET RR = RR + 1
235  REM *** O QUE SERA DESENHADO ***
240  ON RR GOTO 245,270,285,305,320,335,350,365
245  REM *** DESENHA A CABECA ***
250  PLOT 19,12: PLOT 20,12: PLOT 21,12
255  PLOT 19,13: PLOT 21,13: PLOT 19,14
260  PLOT 20,14: PLOT 21,14
265  GOTO 145
270  REM *** DESENHA O PESCOCO ***
275  PLOT 20,15
280  GOTO 145
285  REM *** DESENHA O CORPO ***
290  HLIN 18,22 AT 16: HLIN 18,22 AT 26
295  VLIN 16,26 AT 18: VLIN 16,26 AT 22
300  GOTO 145
305  REM *** DESENHA O BRACO ESQUERDO ***
310  HLIN 14,17 AT 17
315  GOTO 145
320  REM ***DESENHA O BRACO DIREITO ***
325  HLIN 23,26 AT 17
330  GOTO 145
335  REM *** DESENHA A PERNA ESQUERDA ***
340  VLIN 27,36 AT 19
345  GOTO 145
350  REM *** DESENHA A PERNA DIREITA ***
355  VLIN 27,36 AT 21
360  GOTO 145
365  REM *** ENFORCA E ESCREVE HA! ***
370  COLOR= 3: VLIN 5,13 AT 24: VLIN 5,13 AT 27
375  VLIN 5,13 AT 29: VLIN 5,13 AT 32: VLIN 5,10 AT 34
380  PLOT 25,9: PLOT 26,9: PLOT 30,5: PLOT 31,5:
385  PLOT 30,9: PLOT 31,9: PLOT 34,13
390  PRINT  CHR$ (7)
395  VTAB 21: HTAB 1: PRINT  SPC( 255): VTAB 21
400  HTAB 1: PRINT "VOCE PERDEU"
405  PRINT "A PALAVRA ERA ";P$(P)
410  PRINT "VOCE QUER JOGAR OUTRA VEZ ? (S/N)"
415  VTAB 21: HTAB 39: GET QW$
420  IF QW$ <  > "S" AND QW$ <  > "N" THEN  GOTO 415
425  IF QW$ = "S" THEN  RUN
430  TEXT : HOME : END
435  REM  *** VOCE ACERTOU ***
440  PRINT  CHR$ (7)
445  VTAB 21: HTAB 1: PRINT  SPC( 255): VTAB 21
450  HTAB 1: PRINT "VOCE ACERTOU"
455  PRINT "A PALAVRA E ";P$(P)
460  PRINT "VOCE QUER JOGAR OUTRA VEZ ? (S/N)"
465  VTAB 21: HTAB 39: GET QW$
470  IF QW$ <  > "S" AND QW$ <  > "N" THEN  GOTO 465
475  IF QW$ = "S" THEN  RUN
480  TEXT : HOME : END
485  REM   *** QUAL A PALAVRA ***
490  LET P =  INT ( RND (1) * 11)
495  LET P$(0) = "MACACO":P$(1) = "ELEFANTE"
500  LET P$(2) = "DICIONARIO":P$(3) = "COMPUTADOR"
505  LET P$(4) = "IMAGINACAO"
510  LET P$(5) = "TELEFONE":P$(6) = "GEOGRAFIA"
515  LET P$(7) = "AVENIDA":P$(8) = "ARGENTINA"
520  LET P$(9) = "MARGARIDA":P$(10) = "TRANSITO"
525  TEXT : HOME : GR : POKE 34,24
530  VTAB 22:A$ = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
532  IF  LEN (P$(P)) > 10 THEN  STOP
535  PRINT ">";: FOR I = 1 TO  LEN (P$(P))
540  PRINT "-";: NEXT : PRINT "<": PRINT A$
545  COLOR= 3
550  VLIN 10,39 AT 5
555  HLIN 5,20 AT 9
560  PLOT 6,16: PLOT 7,15: PLOT 8,14: PLOT 9,13
565  PLOT 10,12: PLOT 11,11: PLOT 12,10
570  PLOT 4,39: PLOT 6,39
575  PLOT 20,10: PLOT 20,11
580  COLOR= 2
585  RETURN
590  RETURN
595  PRINT  CHR$ (7): VTAB 22: HTAB 13
600  PRINT "ESTA LETRA JA FOI TECLADA"
605  FOR ERT = 1 TO 800: NEXT : HTAB 13: VTAB 22
610  VTAB 22: PRINT  SPC( 25): GOTO 145
615  VTAB 22
620  HTAB I + 1: PRINT  MID$ (P$(P),I,1):C = C + 1
625  RETURN
```

IBM PC

Em 1981, a International Business Machines entrou no mercado dos microcomputadores com o IBM PC (Personal Computer), que logo alcançou grande popularidade, sobretudo em aplicações comerciais.

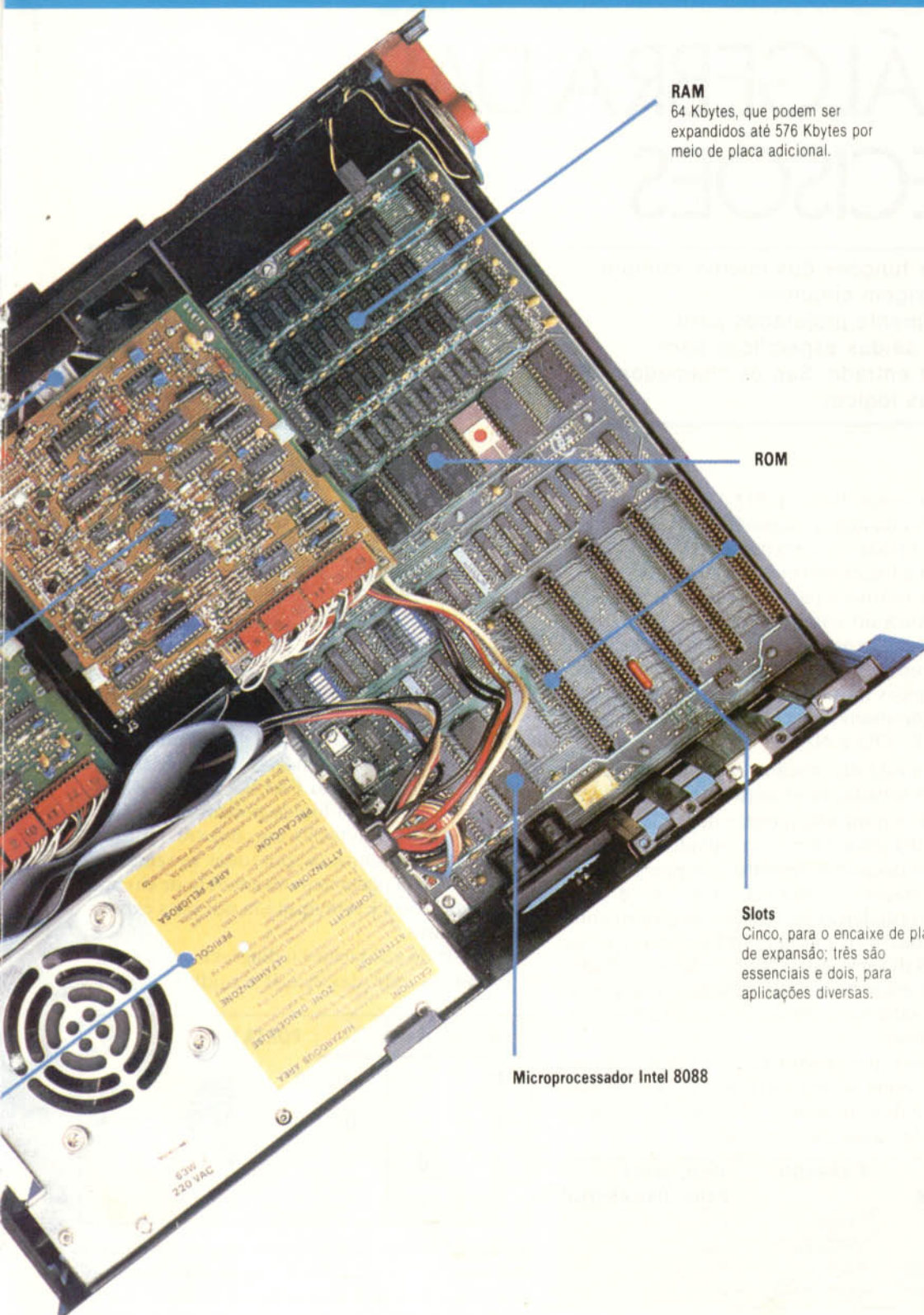
No Brasil e em outros países, multiplicou-se a fabricação de compatíveis do IBM PC. Hoje essa máquina conta com a maior biblioteca de software disponível para qualquer micro: processadores de texto, planilhas eletrônicas, bancos de dados, geradores de gráficos e softwares integrados que combinam aplicações. Isso é possível graças a seu rápido sistema de endereçamento de 16 bits, que permite aproveitar melhor a memória e acessar mais periféricos.

O PC, máquina sólida e confiável, tem suporte técnico garantido pela IBM, a maior fabricante mundial de computadores de médio e grande porte. Seu desenho é funcional, com a CPU, o monitor e o teclado independentes; o usuário os posiciona da forma mais conveniente.

O modelo básico tem 64 Kbytes de RAM. Nele, a execução de programas mais complexos exige placas para expansão de memória, controle de vídeo e de unidades de disco e geração de gráficos.

Unidades de disco
Duas unidades de disquete flexível, de 5 1/4", com dupla face e dupla densidade e capacidade para 360 Kbytes por disco.

Placas de acionamento dos discos

Fonte de alimentação
Chaveada, de 63 watts, 110/220 VAC. Tem ventilador interno, que impede o superaquecimento.

RAM
64 Kbytes, que podem ser expandidos até 576 Kbytes por meio de placa adicional.

ROM

Slots
Cinco, para o encaixe de placas de expansão; três são essenciais e dois, para aplicações diversas.

Microprocessador Intel 8088

IBM PC

MICROPROCESSADOR

Processador central: Intel 8088
Co-processador aritmético: 8087

CLOCK

4,7 MHz

MEMÓRIA

40 K de ROM e 64 K de RAM, expansíveis até 576 Kbytes.

SISTEMA OPERACIONAL

PC-DOS, MS/DOS, CP/M-86, UNIX

VÍDEO

Monitor monocromático em fósforo verde, de 12", com capacidade multinível (várias tonalidades) e vídeo inverso. Dotado de máscara anti-reflexo. Monitor colorido opcional. Alta definição gráfica. No modo texto, a tela apresenta 25 linhas de 80 colunas.

TECLADO

Independente do gabinete da CPU, ao qual se conecta por cabo. Tem inclinação ajustável e é do tipo mecânico, com repetição automática. Possui 82 teclas, sendo dez de função, programáveis. Teclado numérico separado, tipo calculadora, e oito teclas para movimentação do cursor.

LINGUAGENS

BASIC e ASSEMBLER residentes. Diversas outras disponíveis em disco: COBOL, FORTRAN, PASCAL etc.

PERIFÉRICOS

Impressora, disco rígido, joystick, caneta óptica, tablete gráfico, sintetizador de som e de voz, mouse, recursos para teleprocessamento.

DOCUMENTAÇÃO

Completa e detalhada, incluindo manuais de operação, de linguagem BASIC e do sistema operacional DOS.

A ÁLGEBRA DAS DECISÕES

Algumas funções dos micros, como a soma, exigem circuitos especialmente projetados para produzir saídas específicas para qualquer entrada. São os chamados "circuitos lógicos".

O inglês George Boole (1815-1864) foi um dos promotores da lógica matemática. A álgebra por ele desenvolvida — ramo da matemática relacionado com a lógica do verdadeiro ou falso — forma a base teórica a partir da qual é estruturada a arquitetura do computador.

Os conceitos e as regras da lógica booleana são de fácil compreensão. Baseiam-se em três operações lógicas intimamente relacionadas com o modo como usamos, na linguagem cotidiana, as palavras E, OU e NÃO.

Observe esta afirmação: "Se o tempo estiver bom E for sábado, Júlio sairá para um passeio".

Se Júlio vai ou não passear depende de duas coisas: o dia estar bom e ser sábado. Para chegar a uma decisão a respeito, ele precisa saber se as afirmações "o dia está bom" e "é sábado" são verdadeiras ou falsas. Há quatro combinações possíveis (o dia estar bom mas não ser sábado, o dia não estar bom e não ser sábado, o dia não estar bom e ser sábado e o dia estar bom e ser sábado) e apenas uma resultará no passeio de Júlio.

O quadro que mostra todas as combinações possíveis numa série de afirmações é chamado de "tabela de validação". No caso do E lógico, a tabela de validação seria esta:

O DIA ESTÁ BOM	É SÁBADO	JÚLIO SAIRÁ PARA UM PASSEIO
FALSO	FALSO	FALSO
FALSO	VERDADEIRO	FALSO
VERDADEIRO	FALSO	FALSO
VERDADEIRO	VERDADEIRO	VERDADEIRO

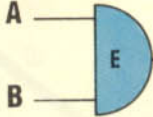
Pode-se realizar um processo semelhante para ilustrar a função da operação lógica OU.

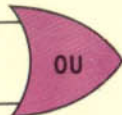
Considere esta afirmação: "Se Pedro OU Paulo puderem ir ao jogo, João também irá".

Uma de duas condições determina se João irá ou não ao jogo: Pedro poder ir ou Paulo poder ir. Como há duas condições — cada uma das quais podendo ser verdadeira ou falsa —, a tabela de validação comporta quatro combinações e somente num dos casos o sujeito não concretizará a ação.

PEDRO PODE IR	PAULO PODE IR	JOÃO IRÁ AO JOGO
FALSO	FALSO	FALSO
FALSO	VERDADEIRO	VERDADEIRO
VERDADEIRO	FALSO	VERDADEIRO
VERDADEIRO	VERDADEIRO	VERDADEIRO

A terceira operação lógica, NÃO, executa uma função muito simples.

Considere esta afirmação: "Se NÃO estiver fazendo frio, irei ao cinema".

Desta vez, a única condição a considerar é se está frio. Ela pode ser verdadeira ou falsa; assim, há apenas duas condições possíveis:

ESTÁ FRIO	IREI AO CINEMA
FALSO	VERDADEIRO
VERDADEIRO	FALSO

Portas lógicas

Os dispositivos eletrônicos simples que formam os circuitos lógicos de computador são chamados de "portas lógicas". Essas portas funcionam pela representação da condição VERDADEIRO pelo bit 1 e da condição FALSO pelo bit 0. Assim, para cada porta lógica podemos construir uma tabela de validação que mostre todas as combinações de entrada com todas as saídas resultantes. Cada porta tem um símbolo de circuito a ela associado e é representada por uma expressão booleana.

A tabela de validação e o diagrama para a porta E com as entradas A e B e saída C é:

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

A função da porta E pode ser descrita em palavras como: "A saída será 1 se ambas as entradas forem 1, e será 0 nos demais casos". A notação booleana para essa saída de uma porta E é A·B (A vezes B).

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

A porta OU descreve-se por esta afirmação: "A saída será 1 se qualquer uma ou ambas as entradas forem 1". A expressão booleana para a saída de uma porta OU é A + B (A mais B).

Ao contrário de E e OU, a porta NÃO tem apenas uma entrada e uma saída. A tabela de validação é a mais simples das três:

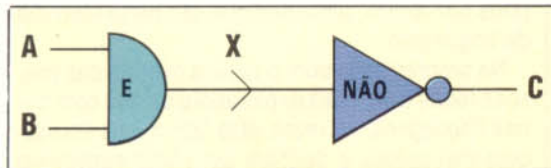
A	B
0	1
1	0

PORTA NÃO

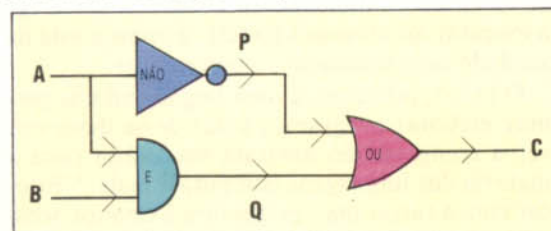
A — NÃO — B

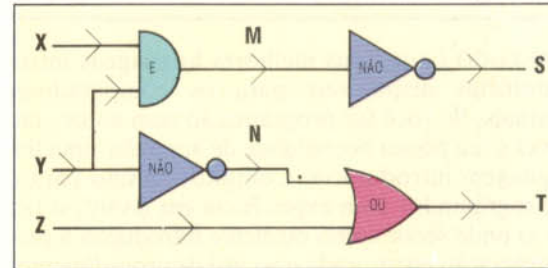
Em palavras, a porta NÃO é expressa como: "A saída será o oposto da entrada". A expressão booleana para a entrada de uma porta NÃO é $\bar{A}$.

Combinando as portas lógicas

É possível unir portas lógicas para formar circuitos lógicos combinados e seqüenciais. Estes, por sua vez, podem ser arranjados para produzir a arquitetura do computador. Qualquer circuito lógico é representado por uma tabela de validação que descreve qual saída esperar para qualquer combinação possível de entradas. Observe este circuito lógico:

Nele, há duas entradas (A e B) e uma saída (C). Para auxiliar o estudante de lógica booleana a construir a tabela de validação para o circuito, a saída da primeira porta foi denominada X. Como existem duas entradas, há quatro combinações de entradas possíveis.

A	B	X	C
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

A saída da porta E (ou seja, X) é realizada pela porta NÃO para produzir a saída final (C). Observe, no circuito mais complexo apresentado abaixo, que, como há duas entradas, o número de combinações possíveis de entrada ainda é quatro. A segunda metade da tabela de validação (P, Q e C) corresponde à reordenação e parte de uma tabela de validação da porta OU.

A	B	P	Q	C
0	0	1	0	1
0	1	1	0	1
1	0	0	0	0
1	1	0	1	1

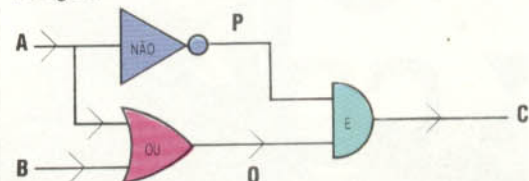
O uso das tabelas de validação não se limita a dois circuitos de entrada e um de saída; pode ser ampliado para qualquer circuito, como os de três entradas e duas saídas:

Como há três entradas neste circuito, devem-se considerar oito combinações possíveis:

X	Y	Z	M	N	S	T
0	0	0	0	1	1	1
0	0	1	0	1	1	1
0	1	0	0	0	1	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	0	0	0
1	1	1	1	0	0	1

EXERCÍCIO 1

1) Construa uma tabela de validação para esta situação: "Jaime pode dirigir um carro se tiver passado em seu exame OU se estiver com um motorista habilitado".

2) Construa uma tabela de validação para esta situação: "Um programa pode ser carregado num computador se houver um gravador cassete OU uma unidade de discos E se o programa NÃO for escrito para ser processado em computador diferente".

3) Construa uma tabela de validação para este circuito lógico:

Respostas na próxima matéria desta seção.

OS PASSOS DA TARTARUGA

O LOGO é considerado a linguagem ideal para desenvolver o raciocínio lógico. Descrevemos aqui a utilização de seus recursos gráficos para obter formas variadas com um mínimo de comandos.

O LOGO é uma das melhores linguagens introdutórias disponíveis para os computadores atuais. Se você faz programação com BASIC, talvez sinta pouca necessidade de aprender uma linguagem introdutória. Contudo, mesmo para o programador com experiência em BASIC, o LOGO pode servir como excelente introdução à programação estruturada e ao uso de procedimentos modulares, em vez de instruções isoladas.

Além disso, ele é encontrado em cartucho ou cassete para a maior parte dos microcomputadores. No Brasil há uma versão LOGO em português (MLOGO), cujos comandos são, por exemplo, DIREITA, FRENTE, REPITA etc. Ela pode ser executada em qualquer micro compatível com o Apple que disponha de um mínimo de 64 Kbytes de memória. Por isso, será usada em todos os nossos exemplos de programas.

Mesmo crianças pequenas (de quatro ou cinco anos) podem ser ajudadas pelo LOGO em seu aprendizado, por meio da "tartaruga", um robô mecânico ligado ao microcomputador. As tartarugas foram projetadas sobretudo para uso escolar e têm princípio de funcionamento simples. A tartaruga é equipada com duas rodas e uma caneta na parte inferior ("barriga"). A criança a instrui para que se mova sobre uma cartolina e desenhe ou não uma linha (o "rastro da tartaruga") conforme se desloca.

Utilizando os comandos LOGO, a criança executa desenhos, instruindo o robô sobre o modo de formar ângulos e unir linhas. Incentivada a planejar os movimentos da tartaruga, descobre os elementos da geometria básica. Essa iniciativa individual constitui o cerne do método LOGO, fundado na idéia de que as lições aprendidas heuristicamente (por tentativa e erro, na prática) são mais bem assimiladas do que as mostradas em teoria e exemplos.

As crianças maiores (a partir de nove ou dez anos) usam a tartaruga na tela do computador, desenhando figuras complexas por meio de procedimentos modulares. Ao ensinar a tartaruga a executar movimentos na tela, a criança está na verdade programando o computador.

O LOGO, portanto, é uma linguagem que permite elaborar programas antes de se desenvolver a compreensão abstrata necessária para a maioria das linguagens computacionais. "Brincar com a tartaruga" possibilita à criança acostumar-se a controlar o computador.

O criador do LOGO, Seymour Papert, numa conferência, em 1983. Papert está associado com o LOGO Computer Systems Inc. (LCSI), que fornece programas para o Sinclair Spectrum, os computadores Atari e outros.

Abordagem exploradora

O LOGO tem duas características fundamentais que fazem dele uma poderosa linguagem educacional.

Ele é interativo: quando você digita um comando, os resultados surgem de imediato na tela. Isso significa que é fácil fazer progresso (em particular, para crianças e principiantes), pois você confere seus resultados a cada passo.

A segunda característica vital é que o LOGO tem recursos para expansão: operações completas são manipuladas por procedimentos, ou seja, listas de instruções elementares em LOGO. Assim que se define um procedimento como um conjunto de instruções, seu nome assume a condição de um novo comando LOGO. Daí em diante, o procedimento inteiro pode ser repetido, bastando para isso digitar seu nome. Dessa forma, você pode criar seus próprios comandos, além dos que são parte inerente da linguagem.

Na programação com o LOGO, a maioria das pessoas tende a ser mais exploradora do que com outras linguagens. Às vezes, elas fazem uma abordagem meticulosa e definem um plano específico desde o começo. Em outras ocasiões, começam com um problema central e escrevem o procedimento para solucioná-lo, montando depois um programa em torno desse procedimento. É possível abordar a programação LOGO de uma maneira mais flexível, porque, em geral, há diversos meios de se chegar a um resultado.

Piaget, Minsky, Papert

O LOGO resultou do LISP, linguagem de inteligência artificial inventada no começo da década de 1960 para ajudar os computadores a trabalhar com estruturas de dados complexas. O nome LISP deriva de ser essa linguagem dedicada ao processamento de listas (LISt Processing). Sua estrutura básica de dados é uma lista, e não um string de caracteres (uma seqüência deles tratados como uma unidade) ou uma matriz numérica, como no BASIC. As funções básicas do LISP manipulam os elementos da lista, que podem ser simples símbolos ou até mesmo outras listas. A vantagem dessa abordagem é que ela processa mais facilmente os dados não numéricos

O LISP baseia-se no princípio da recorrência (recursion), que implica uma série de rotinas repetidas dentro de um programa. O resultado de cada repetição depende do resultado da repetição anterior. No caso do LISP, o item a definir é sempre uma lista.

Essas não são características acidentais do LISP: têm suas origens em pesquisas da linguagem natural e da inteligência humana.

Contudo, não se aprende facilmente o LISP, e em 1968 um grupo de pessoas associadas ao Instituto de Tecnologia de Massachusetts (MIT, Massachusetts Institute of Technology) propôs-se criar uma linguagem para crianças nele baseada.

O líder do grupo do MIT era Seymour Papert, que já havia dedicado vários anos ao estudo do desenvolvimento cognitivo infantil, com Jean Piaget (1896-1980), o mais importante psicólogo educacional de sua geração. Passando para o MIT, começou a trabalhar ao lado de Marvin Minsky, especialista em inteligência artificial. Em seu trabalho com o LOGO, Papert tentou juntar as idéias dos dois cientistas, unindo teorias do aprendizado e da inteligência artificial.

As pesquisas com o LOGO continuaram durante os anos 70, e formaram-se outros grupos dispostos a fazer experiências usando a nova linguagem. Todo esse trabalho desenvolveu-se nos departamentos de pesquisa de várias universidades, com computadores de médio e grande porte. Foi só com a chegada dos microcomputadores que o LOGO se tornou disponível para o público em geral. Essa sofisticada linguagem necessita de muita memória, tanto para o programa quanto para o espaço de trabalho. Os interpretadores LOGO encontrados nos micros requerem uma memória em torno de 30 Kbytes, e outros 8 Kbytes ou mais para a exibição dos gráficos. Tudo isso, antes de se começar a programação.

Assim, embora fosse possível implementar interpretadores BASIC simples em micros de uso doméstico a partir do momento em que eles foram comercializados, só depois que os micros com mais de 48 Kbytes de RAM passaram a ser largamente usados é que o LOGO em micros se tornou viável. Em 1980, a publicação de *Mindstorms (LOGO — Computadores e educação)*, de Seymour Papert, tirou o LOGO dos departamentos de pesquisa, atraindo a atenção de um grupo muito maior de pessoas. Nesse livro desenvolve-se uma visão de como os computadores podem ser usados na educação. A obra re-

Versões autorizadas
Versões completas e bem documentadas do LOGO para o Commodore 64 (Terrapin-MIT), Sinclair Spectrum e os computadores Atari (LCSI). No Brasil existe a versão MLOGO, em português.

Geometria e lógica

Quem pode se beneficiar com o aprendizado da programação em LOGO? Muitas pessoas, e até mesmo programadores experientes:

- Principiantes em computação ou programação;
- Pessoas que gostam de entreter-se com computadores e acham que usá-los deve ser uma atividade divertida;
- Aqueles que estão decepcionados por não terem adquirido um domínio suficiente em outra linguagem de programação;
- Quem se mostra interessado em aprender, ensinar e desenvolver o raciocínio lógico e matemático; e
- Usuários que desejam penetrar em áreas mais avançadas da computação, especialmente aquelas que estão relacionadas com o estudo da inteligência artificial.

Assim como o BASIC e a programação em código de máquina, o LOGO não é para ser usado por qualquer um. Especificamente, pode não ser a melhor linguagem para:

- Quem pensa que usar o computador é um "trabalho". Algumas linguagens são planejadas para trabalhar, como cavalos que puxam carroças. Mas você dificilmente escolheria um cavalo de carroça para montar e passear no campo; e
- O usuário que precisa ou espera grande velocidade no processamento das instruções pelo computador. O LOGO usa muita memória e funciona devagar na atual geração de microcomputadores. (Em programas comparáveis, o LOGO funciona com metade da velocidade do BASIC.)

No entanto, mesmo para esses grupos, o conhecimento do LOGO pode ser de grande valia para indicar a solução lógica de um problema e prepará-lo para ser programado em outra linguagem.

Vem aí...

... a tartaruga!

sùltou da síntese de três conjuntos de idéias: teorias do desenvolvimento cognitivo, inteligência artificial e o movimento educacional que visa ao ensino centrado no aluno. Papert quer ver crianças programando computadores, e não computadores programando crianças — o que, segundo ele, acontece na maioria dos casos de "ensino com auxílio de computador".

O livro prevê o surgimento de uma nova "cultura do computador", na qual muitas idéias formais, anteriormente consideradas acima da capacidade das crianças, serão por elas dominadas com facilidade, devido à maneira pela qual elas usam o computador. Essa exploração de idéias — ativa e cooperativa (numa relação aluno-aluno e aluno-professor) — constitui a filosofia LOGO, na qual se fundamenta o uso dessa linguagem na educação.

Simplificação

Abreviações dos comandos apresentados nesta matéria:

FRENTE	FR
VOLTE	VO
DIREITA	DI
ESQUERDA	ES
SEMCOR	SC
COLORIDO	CL
MOSTRE	MO

A tartaruga

Depois de carregado, o LOGO fica em modo "imediato" e está pronto para receber e executar comandos. Na maioria das versões, esses comandos devem ser digitados com letras maiúsculas. Emita DESENHE e você verá que a tela se divide em duas partes (modo de tela dividida). A ala superior fica reservada aos gráficos e ocupa a maior parte da tela. No centro aparece a tartaruga, representada por um pequeno triângulo. A parte inferior cabe ao texto, e no momento apresentará um "?".

A tartaruga é um objeto com o qual podemos nos comunicar, transmitindo-lhe comandos. Quando se pensa na tartaruga como um objeto, fica mais fácil entender como se faz programação com ela. As coisas mais importantes a considerar são a posição da tartaruga, sua direção, e se a caneta que ela leva está virada para baixo (caso em que traça uma linha ao mover-se) ou para cima (move-se sem deixar traço). Emitindo-se o comando DESENHE, a tartaruga fica posicionada no centro da tela, virada para o alto e com a caneta abaixada. Podemos então lhe dar um comando como este:

```
FRENTE 40
```

No caso, a tartaruga subirá quarenta unidades ("passos de tartaruga") na tela, traçando uma linha. FRENTE é um comando para a tartaruga, e o número 40 é seu dado de entrada. Alguns comandos necessitam de dados, ao passo que outros não. DESENHE, por exemplo, não precisa de dados.

Um segundo comando para a tartaruga é VOLTE. VOLTE 10 instrui a tartaruga para que volte dez unidades. Portanto, FRENTE e VOLTE mudam a posição da tartaruga na tela. DIREITA e ESQUERDA, por outro lado, não mudam a posição da tartaruga, mas fazem-na girar — isto é, mudam sua direção. Estes dois últimos requerem um ângulo entre 0 e 360 graus como dado.

Experimente usar esses comandos desenhando algumas formas simples. Veja o que acontece se você instrui a tartaruga para mover-se a uma distância maior do que permite o tamanho da tela ou se usa números negativos como dados. Para começar outra vez com a tela limpa, digite DESENHE.

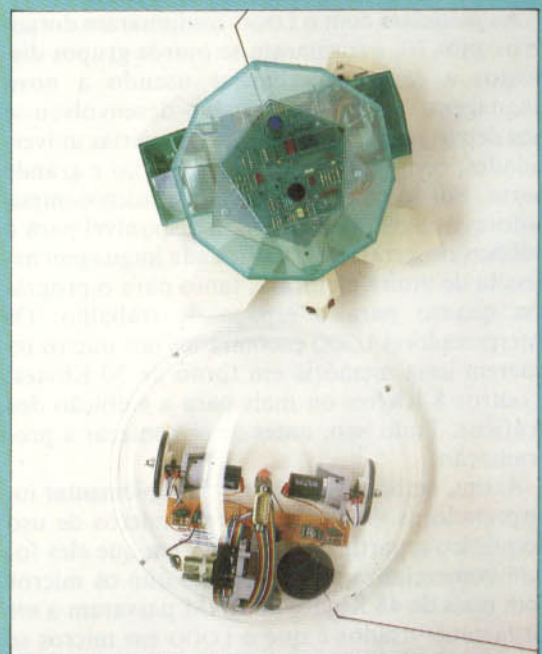
Quando testar os comandos, verá que a tartaruga pode entrar na outra seção da tela, e assim ficar "atrás" do texto, de modo a não ser vista. Estes comandos podem ajudá-lo:

TODATELA — permite que a área total da tela seja usada para os gráficos;
ESCREVA — remove todos os gráficos, deixando apenas o texto dos procedimentos;
MEIATELA — devolve o modo de tela dividida para gráficos e textos;
SEMCOR — faz com que a tartaruga se mova sem traçar uma linha;

Conheça a tartaruga

Em LOGO, uma tartaruga é um robô que desenha. Ela possui rodas, controladas por motores graduais, e uma caneta retrátil. Pode ser instruída a fim de andar para a frente, para trás, para a direita e para a esquerda, e pode levantar ou abaixar sua caneta. Abaixada, a caneta desenha sobre a superfície em que a tartaruga está colocada.

Quando começaram a ser produzidas, as tartarugas tinham a forma de cúpula, como a Edinburgh *(foto de cima)*, e eram controladas pelo teclado do computador, ao qual se ligavam por um cabo paralelo. As tartarugas mais modernas funcionam por controle remoto. Existe hoje uma versão da tartaruga Edinburgh controlada por rádio. A Valiant Turtle, um robô em forma de tartaruga *(foto de baixo)*, tem uma conexão infravermelha com o computador.

Por extensão, o nome "tartaruga" também é usado para referir-se ao cursor que desenha na tela do computador, em LOGO.

COLORIDO — determina que a tartaruga deixe uma trilha enquanto se move.

Os comandos que descrevemos até agora correspondem a instruções para a tartaruga fazer o que se deseja. Mas você também pode usar o LOGO para obter informações da tartaruga. Sua direção é medida em graus, no sentido horário. A direção 0 indica que a tartaruga está virada diretamente para cima. Para descobrir a direção da tartaruga, digite:

```
MOSTRE DIRECAO
```

A posição da tartaruga na tela é definida em termos de um sistema de coordenadas, com origem no centro da tela — isto é, a tartaruga parte de um ponto onde as coordenadas *x* e *y* são iguais a 0. Você pode determinar a posição da tartaruga a qualquer momento, digitando:

```
MOSTRE CORX MOSTRE CORY
```

Teste os comandos desenhando uma figura, e então verifique a posição da tartaruga e a direção para a qual está voltada. Use esses dados para fazer a tartaruga voltar ao ponto de partida.

Mensagens de erro podem aparecer na área do texto durante suas experiências com os comandos LOGO. Se isso não acontecer, erre deliberadamente para ver o que ocorre. Digite, por exemplo:

```
FRENTE50
```

Você verá a mensagem:

```
NAO APRENDI O QUE SIGNIFICA FRENTE50
```

A razão disso é que o LOGO exige um espaço entre o comando FRENTE e o dado 50, para evitar confusão com um hipotético comando chamado FRENTE50. Você também pode obter uma mensagem de erro se digitar um comando em letras minúsculas.

O LOGO é equipado com um editor de linhas que lhe permite corrigir as instruções, se você notar um erro antes de pressionar [RETURN]. Use as teclas de movimentação do cursor para percorrer a linha até o texto incorreto. Para inserir os caracteres, basta digitá-los — o texto à direita do erro será automaticamente movimentado para acomodar os caracteres extras desejados. A tecla [ESCAPE] apaga o caractere à esquerda do cursor.

Quando a linha está correta, tecla-se [RETURN], o que faz o LOGO aceitar a nova instrução. Se você já pressionou [RETURN] antes de perceber um erro, digite [CONTROL]-P, a fim de conservar a última linha para edição. Essa função é igualmente útil quando você deseja repetir comandos.

Agora podemos tentar algo um pouco mais matemático, como um quadrado. Lembre-se de que um quadrado tem quatro lados iguais e que seus cantos formam ângulos retos (90 graus). Portanto, uma seqüência de comandos como esta produzirá o resultado desejado:

```
FR 50 DI 90
FR 50 DI 90
FR 50 DI 90
FR 50 DI 90
```

Observe que podemos abreviar os comandos e colocar mais de um comando na mesma linha. Por outro lado, você vai encontrar um problema técnico nesse ponto: seu quadrado pode ficar parecido com um retângulo. Isso se deve à desproporção entre as medidas verticais e horizontais do passo da tartaruga. Há um comando LOGO que cuida disso: use .VERTICAL seguido de um número (a definição pré-assumida é de 0.8), para modificar a proporção até que se configure um quadrado perfeito.

Agora, tente fazer os seguintes exercícios: desenhe um triângulo eqüilátero, um pentágono, um hexágono, diversos retângulos, um losango e um paralelogramo.

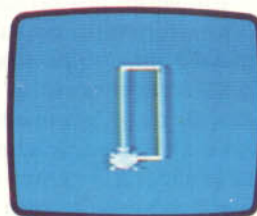
Você pode simplificar alguns dos comandos e reduzir o volume de digitação usando REPITA. Para desenhar de novo o quadrado, digite apenas:

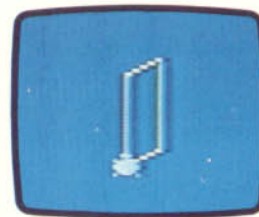
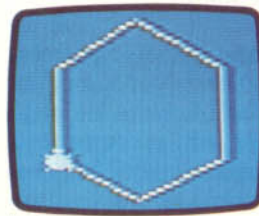
```
REPITA 4 [FR 50 DI 90]
```

REPITA é um comando que exige dois parâmetros. O primeiro é um número que indica quantas vezes o LOGO deve fazer alguma coisa, e o segundo é uma "lista" dos comandos que devem ser obedecidos. Essa lista deve vir sempre entre colchetes. Assim, o exemplo do quadrado diz ao LOGO que ele deve repetir a seqüência FRENTE 50 DIREITA 90 quatro vezes. Agora experimente usar REPITA para simplificar a construção das figuras que você já criou, e veja se pode produzir as estrelas que aparecem nas ilustrações desta matéria.

Triângulo eqüilátero

Retângulo

Exercício 1
Você consegue escrever os procedimentos para criar estas formas? **Respostas na próxima matéria da série.**

Pentágono

Estrela de cinco pontas

Paralelogramo

Hexágono

Estrela de dez pontas

PLANO DE AÇÃO

A boa programação considera aspectos como a finalidade do programa, sua utilização, os formatos de entrada e saída, os formatos dos arquivos, as rotinas e os cálculos especiais, além da estruturação lógica e modular.

Os projetos de programa são vistos pelos que estão envolvidos — analista, programador e usuário — como exercícios formais de solução de problemas. E, em geral, consideram-se apenas os aspectos técnicos — como formatar a tela, tornar os loops mais rápidos, fazer com que o programa caiba na memória, e assim por diante. No entanto, as questões já surgem no primeiro encontro do usuário com o especialista. Com raras exceções, o usuário espera do especialista não apenas a solução mas também a indicação da natureza do problema.

Junta-se a isso a freqüência com que os especialistas acreditam conhecer o problema e a solução antes mesmo de o usuário começar a esboçá-lo. Dessa má comunicação inicial resulta a descrição incompleta das necessidades do usuário. Trabalhar desse modo só pode levar a um sistema insatisfatório, que o usuário apenas aceitaria sob pressão.

Em geral, nos projetos de microcomputação, o programador, o usuário e o analista do sistema são uma só pessoa. Nesse caso, reduzem-se consideravelmente as falhas de comunicação. Apesar disso, quem acumula as funções de usuário e analista deve sempre fazer um esforço para explicar a si mesmo os problemas, as soluções e as necessidades com tanta clareza como se falasse com outra pessoa.

Um exímio aeromodelista, por exemplo, quer armazenar em seu microcomputador, acoplado a gravador cassete, descrições bem precisas dos materiais utilizados em cada modelo por ele construído. Isso lhe permitirá, no futuro, consultar seus arquivos a respeito do tipo de cola ou junta utilizado em modelos anteriores. Assim, o que o projetista teria de obter do usuário seria uma definição clara sobre:

■ **A função do programa.** Pode-se começar com uma definição vaga do objetivo, como, por exemplo: "Deve armazenar os dados de meus modelos". É necessário, porém, melhorar essa definição mediante um insistente questionamento pelo projetista. Assim, torna-se possível especificar exigências do tipo "O programa precisa armazenar minhas descrições do modelo, de sua construção e de seus materiais, à medida que forem digitadas, e exibi-las na tela quando eu teclar o nome do modelo ou alguma característica de sua construção". Isso estabelece com mais clareza as exigências do usuário e indica algumas das tarefas de programação específicas envolvidas (armazenamento de dados, pesquisa, indexação, recuperação, correção).

■ **Como o programa vai ser utilizado.** Alguns detalhes da configuração disponível ficam claros pela descrição da função, mas podem não ser suficientes. Por exemplo, se o usuário trabalhar sem monitor, não haverá interesse por exibição na tela; nesse caso, seria imprescindível uma cópia impressa dos detalhes selecionados.

■ **Qual o aspecto do programa — formatos de entrada e saída.** O programador profissional costuma trabalhar com diagramas pré-impressos que representam a tela em cada fase de entrada e de saída. Para uso pessoal, essa preocupação nem sempre é necessária, embora os gráficos de alta resolução sejam exceção. Os formatos de tela constituem um aspecto muito importante na "interface com o usuário", ou seja, o contato entre usuário e máquina. Merecem, por isso, atenção e discussão especiais, semelhantes às dedicadas aos aspectos ergonômicos mais óbvios da computação, tais como a posição dos teclados e monitores, a altura da mesa etc.

■ **Como o programa deve ser organizado — os formatos do arquivo e do programa.** O usuário pode achar que necessita estocar no mínimo cem descrições de modelo, considerando inútil menos que isso. Por outro lado, pode ser que ele jamais chegue a criar senão uma meia dúzia de modelos padronizados. O tamanho do arquivo de dados tem sérias implicações sobre seu formato e métodos de acesso. Uma pesquisa serial das seis descrições de modelo guardadas em fitas cassete poderia levar uns cinco minutos, e seria aceitável para o usuário. Mas esperar que cem modelos sejam pesquisados estaria totalmente fora de questão. Uma solução seria colocar o programa e o índice das descrições numa fita, e as descrições propriamente ditas em outras vinte fitas, classificadas por tipo de modelo.

O tamanho do programa pode também vir a ser um problema. Em alguns casos, é preciso dividi-lo em vários programas distintos, a fim de que caiba na memória disponível. Por exemplo, se a entrada de textos requer recursos comple-

xos de edição; se o programa é cheio de menus e mensagens longas; se as seções de manipulação de arquivo utilizam rotinas complicadas de pesquisa e indexação.

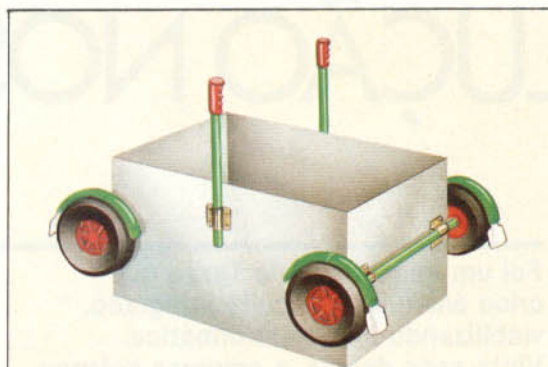
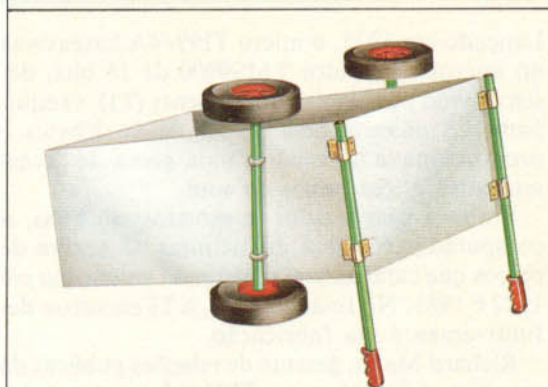
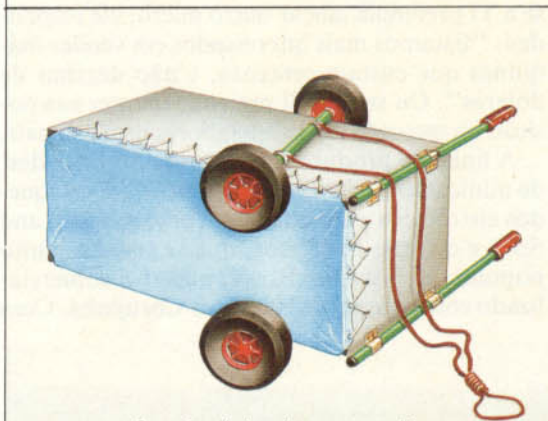
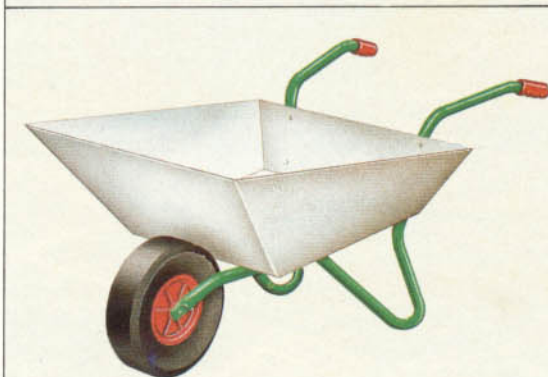
■ **O que o programa deve fazer — rotinas e cálculos especiais.** No exemplo dos aeromodelos não ocorriam problemas desse tipo; em outros casos, porém, são freqüentes. É possível que haja vinte maneiras boas e equivalentes de se realizar uma tarefa específica, mas o usuário poderá, ainda assim, insistir numa única solução. E mesmo que outros métodos sejam melhores, de nada adiantarão se o usuário não quiser utilizá-los.

A lógica necessária

O passo seguinte é a tradução das especificações em linguagem de programação. Uma maneira prática de começar é primeiramente projetar o "diálogo" entre o usuário e o programa; em seguida, os arquivos de dados; e por último as rotinas que controlam todo o processo. Diálogo, no caso, significa troca de informações, e não, simplesmente, a entrada dos detalhes do aeromodelo e a correspondente exibição em tela. Inclui também as mensagens e os menus mostrados pelos programas, e as entradas, os comandos e as escolhas que o usuário deve digitar. Para o programa dos aeromodelos escolhe-se entre o sistema de menus e os comandos interativos. As decisões tomadas terão efeito considerável sobre a estrutura de todo o programa.

O conteúdo e a forma de comunicação devem ser considerados em detalhe, mas há uma recompensa para esse esforço: todos os dados manipulados pelo programa estarão especificados nesse momento. Isso significa que o espaço requerido para armazenamento das mensagens e advertências pode ser calculado; e torna-se possível, assim, projetar os arquivos.

No caso de arquivos longos com grandes blocos de texto, como os do programa do aeromodelo, convém dividi-los em várias fitas. Dessa forma, cada fita pode ser pesquisada com rapidez. Outro esforço útil é comprimir os dados numa codificação algorítmica, antes que sejam transcritos em fita, e, depois, decodificá-los na leitura. (Algoritmo é um conjunto de instruções que descreve um processo em termos de outros processos, básicos ou anteriormente definidos. São comuns tanto em matemática e estatística quanto em receitas de cozinha ou de tricô.)

A essa altura, as funções necessárias estarão definidas. Haverá rotinas para alterar os dados já existentes e também para arquivar e classificar as novas entradas, aceitar os nomes de componentes, bem como pesquisar e exibir na tela as descrições. A essas rotinas cabe ainda a atualização de todos os índices utilizados pelo sistema. O programa deve incluir uma rotina de verificação da consistência dos dados introduzidos, a fim de rejeitar os que não sejam válidos. Todas essas opções serão apresentadas ao usuário sob a forma de menus.

O que o analista pensou que o usuário precisava

O que o programador pensou que o analista tinha dito

O que eles afinal venderam para o usuário

O que o usuário de fato queria

Descrever, definir, projetar
Se uma pouco atenta equipe de desenvolvimento de software — usuários, analistas e programadores — se propusesse a resolver o problema do transporte de cargas pesadas em jardins, é bem possível que o resultado se assemelhasse ao da ilustração. Má comunicação — entre especialistas — ainda é o maior problema a desafiar as equipes de projeto.

REVOLUÇÃO NOS CIRCUITOS

O presidente
J. Fred Bucy passou a desempenhar a função de presidente da Texas Instruments em abril de 1976.

Foi um engenheiro da Texas que criou em 1958 o circuito integrado, viabilizando a microinformática. Vinte anos depois, a empresa colocou no mercado o primeiro computador pessoal da história.

Lançado em 1978, o micro TI99/4A baseava-se no microprocessador TMS9900 de 16 bits, desenvolvido pela Texas Instruments (TI). O equipamento possuía uma RAM de 16 Kbytes e proporcionava ao usuário uma gama de dezesseis cores e três canais de som.

Embora suas vendas se mostrassem boas, o computador foi uma das vítimas da guerra de preços que caracterizou o mercado americano em 1982 e 1983. No final de 1983, a TI encerrou definitivamente sua fabricação.

Richard Mann, gerente de relações públicas da empresa, declarou que o TI99/4A era pouco lucrativo e "acabamos perdendo centenas de milhões de dólares". Quando lhe foi perguntado se a TI pretendia lançar outro micro, ele respondeu: "Estamos mais interessados em vender máquinas que custam centenas, e não dezenas de dólares". Ou seja: a TI pretende manter sua posição no mercado de computadores profissionais.

A linha de produtos da empresa abrange desde minicomputadores e sismógrafos até brinquedos eletrônicos para crianças, como o Speak and Spell e o Little Professor, que se tornou muito popular, inclusive no Brasil, onde foi comercializado com o nome de Professor Corujinha. Com quinze fábricas espalhadas pelo mundo, em meados da década de 80 faturava mais de 4 bilhões de dólares por ano.

A fábrica da TI
Grande parte dos equipamentos da Texas Instruments são produzidos neste moderno complexo industrial em Dallas.

Quando foi fundada, em 1930, pelos cientistas John Karchner e Eugen McDermott, seu nome era Geophysical Service Incorporated (GSI). Karchner realizara pesquisas sobre a utilização de ondas sonoras para o cálculo da profundidade das camadas geológicas. Como pretendia vender essa sua idéia às companhias petrolíferas, resolveu abrir sua própria empresa em Dallas, no Texas.

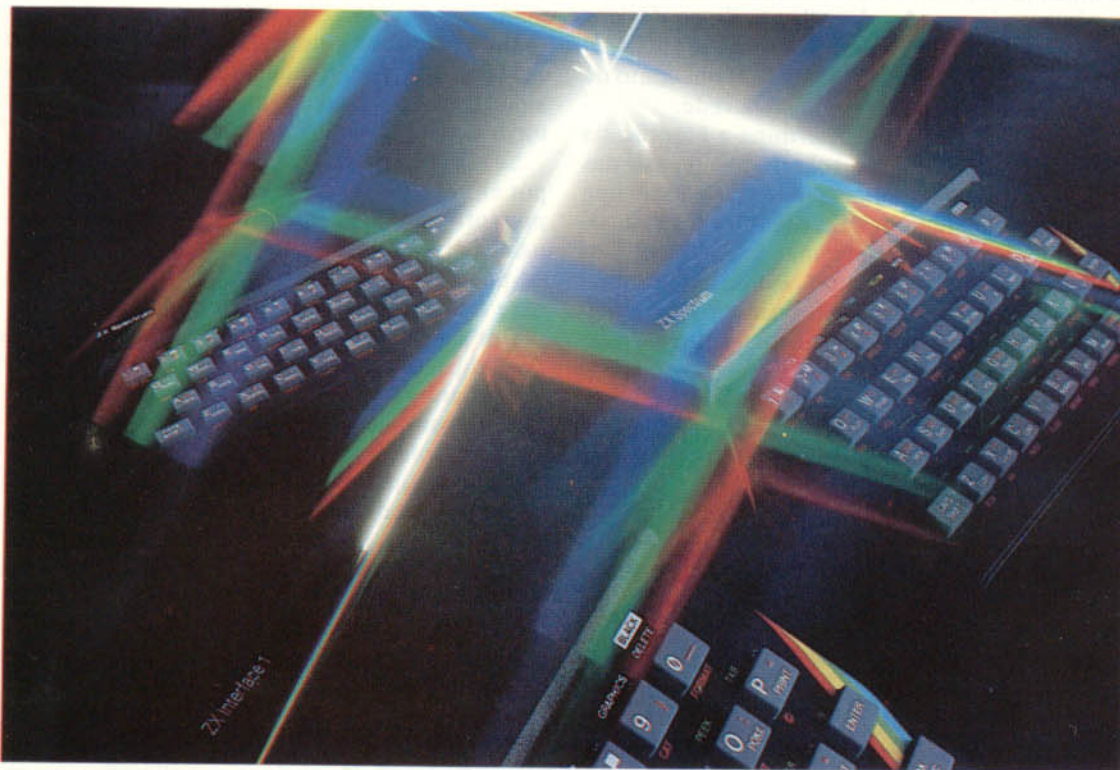
Os anos 30 foram de contínua prosperidade para a GSI, graças ao desenvolvimento de técnicas e equipamentos de prospecção geológica. Durante a Segunda Guerra Mundial (1939-1945), esses equipamentos revelaram-se muito úteis na detecção de submarinos inimigos. Implantaram-se então um laboratório e uma linha de produção especiais na GSI. Em 1951, o setor crescera tanto que se tornou conveniente desligá-lo da GSI. A nova companhia recebeu o nome, hoje consagrado, de Texas Instruments.

No ano seguinte, a TI obteve licença dos Laboratórios Bell para a fabricação dos transistores, que acabavam de ser inventados. O primeiro rádio transistor comercial saiu de suas linhas de produção em 1954. Quatro anos depois, Jack Kilby, um de seus engenheiros, inventou o circuito integrado. Nos anos seguintes, a TI manteve-se na vanguarda das pesquisas nessa área. Kilby participou ainda do desenvolvimento da primeira calculadora eletrônica portátil, lançada em 1967.

O computador profissional da TI foi comercializado em janeiro de 1983. Baseado no processador 8088, ele pode rodar o CPM-36 e o MS-DOS. Nesse mesmo ano, a TI apresentou seu computador profissional portátil e o micro CC40, do tamanho de uma calculadora de bolso.

Na enorme lista dos componentes eletrônicos produzidos pela TI, destacam-se os chips RAM de 64 Kbytes. Os chips da Texas Instruments são encontrados em inúmeros microcomputadores. Seguindo a tendência atual, a TI também desenvolveu sua linha de processadores de 16 bits, como o TMS9900. Ao contrário de outros processadores, o TMS9900 não possui registros internos. Estes são incluídos numa memória auxiliar externa à CPU.

Segundo Richard Mann, "o TMS9900 estava um pouco à frente de seu tempo. Ele possuía muitos recursos poderosos, mas sua arquitetura não foi bem aproveitada". Posteriormente, a Texas Instruments colocou no mercado outro processador de 16 bits, o TMS99000.

INFORMAÇÃO EM CADEIA

Interface
Mesmo micros simples, como os ingleses da linha Sinclair, podem intercomunicar-se e partilhar recursos, quando equipados com interfaces apropriadas.

As redes não exigem equipamentos complicados. Compartilhando dados e periféricos caros, como impressoras, monta-se a custo compensador um sistema de microcomputadores interligados.

Há alguns requisitos a preencher antes de se conectarem computadores em rede. Eles devem estar próximos o suficiente para ser ligados por cabo — de preferência no mesmo prédio. É necessário, ainda, que haja tráfego de comunicação suficiente para que valha a pena montar a rede. Os usuários devem intercambiar dados muitas vezes por dia, ou partilhar um equipamento caro (tal como impressoras ou unidades de disco) para amortizar o custo da rede. Se apenas um pequeno número de dados passar pela rede, será mais fácil que cada usuário entregue ao outro uma fita ou um disco.

Cada computador tem seu próprio sistema operacional e seu próprio software para se comunicar com outros equipamentos. Conforme o tipo de configuração, todos os terminais de uma rede devem ser computadores do mesmo tipo. Por exemplo: todos IBM PC, ou todos Apple.

Vamos supor que um grupo de pessoas possua cinco computadores e queira compartilhar uma impressora. No caso, cada terminal deve enviar suas informações para a impressora.

O que fazer quando dois ou três terminais se dispõem a imprimir seus textos ao mesmo tempo? E se um terminal tem texto para imprimir mas quer também continuar trabalhando, enquanto a impressora opera?

Para resolver problemas como esses, instala-se um sexto computador, o controlador de impressão. Essa máquina tem como tarefa controlar o fluxo de dados a imprimir, sendo utilizada principalmente para esse fim. Ela armazena os documentos conforme a ordem de prioridade. Uma vez enviado o texto para o controlador de impressão, o terminal fica liberado para executar outras tarefas. É essencial que a rede use a máquina dedicada a controlar o fluxo das informações para a impressora, pois é por meio desse computador extra que as informações podem ser impressas.

Além do controlador de impressão, algumas aplicações de rede necessitam obrigatoriamente de um micro concentrador ou gerenciador de arquivos para lidar com unidades de disco compartilhadas e controlar o fluxo de informações entre os terminais.

O próximo passo: criar a ligação dos terminais. Para isso usa-se um cabo do tipo co-axial ou par trançado, que vai de máquina a máqui-

na. Embora haja várias maneiras de conectar os terminais e o micro concentrador, como a rede em estrela e a rede em anel, o conceito básico é o mesmo.

Montar a rede exige interface apròpriada em todos os terminais. Essa interface pode ser uma simples conexão tipo RS232 ou uma placa que se encaixa num dos slots do micro. Além disso, o micro concentrador deve dispor de periféricos para armazenamento com capacidade suficiente para todo o fluxo de trabalho, assim como de RAM suficiente para gerenciar a rede.

O software utilizado determina a atuação da máquina e isso é válido sobretudo no caso das redes. Um sistema gerenciador da rede, acima do sistema operacional das máquinas, faz a conexão lógica de cada terminal com a rede. Conhecido como "protocolo", ele dá ao terminal concentrador o controle das operações de arquivo e/ou de impressão, e também do fluxo de dados pela rede.

Além de estabelecer essa cadeia de comandos, o protocolo informa a cada terminal que ele está em rede, subordinado a um micro gerenciador, e quantos outros terminais existem na rede. Por fim, o software de rede interage com o protocolo para que os terminais se comuniquem com o restante do sistema.

Experiência compartilhada
Esta escola foi equipada com micros dotados de monitores coloridos, uma unidade de disco rígido e uma impressora, o que fica mais barato do que equipar cada micro com uma unidade de disco e uma impressora em separado.
A velocidade da rede é tal que cada máquina parece acoplada a uma unidade de disco individual, mesmo quando trinta estudantes estão operando ao mesmo tempo. Às vezes os terminais ficam "na fila" quando querem utilizar a impressora. O sistema permite que o professor forneça experiência prática a todos os alunos. No ensino, a comunicação entre terminais é sempre uma grande vantagem.

Uma vez estabelecida a rede, cada terminal deve ter seu programa ou um conjunto de programas para suas próprias aplicações. Esses softwares reconhecem a rede e sabem como se comunicar com ela.

Desenvolvidos especialmente para aplicações em rede, tais programas podem ser executados a partir de um disco no próprio terminal ou por meio do micro concentrador.

Se o terminal estiver processando um texto, por exemplo, e enviando os resultados para a impressora, independentemente dos demais terminais, a única condição necessária ao processador de texto para que ele possa ser processado em rede é a inclusão de protocolos.

Por outro lado, se os terminais **B** e **D** tiverem de compartilhar os mesmos dados e um precisar consultar os resultados do outro, a operação em rede se complica. No caso, o software aplicativo (um processador de texto, uma planilha financeira, um banco de dados ou mesmo um jogo) e o hardware de sistema devem ser capazes de operar em conjunto.

Em outras palavras, a CPU (Unidade Central de Processamento) precisa executar várias tarefas ao mesmo tempo e manipular comunicações simultâneas provenientes de, pelo menos, duas outras CPUs.

Acesso e seleção

As redes locais têm inúmeras aplicações em pequenos e médios empreendimentos, comerciais e industriais. A uma administradora de imóveis, por exemplo, convém que todos os dados relativos a imóveis, condôminos, funcionários, despesas etc. fiquem registrados numa base de dados compartilhada. Essa base pode ser um disco rígido, caso o volume total de dados seja grande o bastante. O micro concentrador fica encarregado de lançar e atualizar os dados. Um segundo micro lança na mesma base os pagamentos em atraso, feitos na caixa da administradora. Outro emite os carnês de pagamento e os relatórios, e um quarto, utilizado pela contabilidade, controla pagamentos e débitos, envia cartas de cobrança e se encarrega da folha de pagamento dos funcionários.

Compartilhando uma impressora e um disco rígido, todos os serviços seriam executados com acesso a dados atualizados.

Por outro lado, pode-se evitar que os dados sejam indevidamente acessados estabelecendo restrições na configuração da rede. Por exemplo, o micro **A** acessa o **B** e o **C**, mas só é acessado pelo **D**; o **B** tem acesso ao **C** mas não ao **A**, e assim por diante. A própria base de dados pode ter setores restritos a este ou àquele micro. Dessa forma, todos trabalham com dados atualizados, mas só os acessam com prévia autorização.

Concentrador
Controla o tráfego da rede. Cada usuário tem sua própria senha para acessar a rede e seus próprios diretórios de arquivos. Há também um índice dos arquivos disponíveis a todos os usuários.

EDITOR-CHEFE

Caixa de sincronização
Gera os sinais de tempo que sincronizam todas as comunicações da rede.

Impressora
Acessada a partir de qualquer terminal da rede e controlada por ROM especial num dos terminais, deixando-o livre para o trabalho normal.

EDITOR DE PRODUÇÃO

Acoplador acústico (modem)
Passa informações por via telefônica da unidade de disco para um terminal remoto ligado à rede.

NÓ DE JUNÇÃO

SUBEDITOR

Terminador
Termina eletricamente as extremidades do cabo de comunicação (bus) para evitar a degradação dos sinais.

Identificação do terminal
Cada terminal tem um número específico (entre 001 e 254), fixado por chaves internas, para se identificar dentro da rede.

Acesso dos terminais
Todo terminal é capaz de enviar para qualquer outro, ou dele receber, mensagens que são mostradas na tela.

REDATOR

REDATOR

Edição em rede

Esta rede está sendo usada para produzir um jornal fictício. Os redatores utilizam terminais com processadores de texto e armazenam os textos na unidade de disco compartilhada. Sempre que desejar, a equipe de produção pode visualizar as telas dos redatores nas telas de seus próprios terminais e produzir o texto final a partir dos arquivos em disco. O micro do editor-chefe controla a rede, de modo que o texto final seja impresso. Arquivos de textos vindos de computadores remotos chegam ao concentrador por um modem.

DADOS NA TELA

Num jogo ocasional ou em cartas comerciais comuns, a qualidade da imagem é fator secundário. Mas, quando se destina à educação ou ao uso empresarial, crescem as vantagens do monitor de vídeo.

O vídeo, um dos periféricos mais importantes do computador, costuma ser o aparelho de TV comum, muitas vezes em preto e branco, comprado especialmente para esse fim. A qualidade assim obtida é aceitável, mas os resultados não são os melhores.

Isso porque o sinal enviado da memória de vídeo passa por vários estágios de codificação e decodificação antes de aparecer na tela. Por melhor que seja a qualidade do circuito em questão, o resultado é sempre imperfeito: as imagens finais muitas vezes saem borradas, difíceis de ler e com um desagradável efeito de tremulação.

O segredo da alta qualidade da imagem está na eliminação dessas distorções de sinal produzidas pelos circuitos de modulação e demodulação; para tanto, basta eliminar esses estágios.

É aí que se encontra a vantagem do monitor. Sem os estágios de codificação e decodificação utilizados na televisão, o monitor é um sistema simplificado capaz de gerar imagens mais nítidas, brilhantes e estáveis.

Como não existem decodificadores de sinal, o monitor não funcionará quando conectado à saída comum de televisão. O sistema exige saída de vídeo própria, que não passe pelo modulador. Para verificar isso, consulte o manual de seu computador.

O processo de gerar imagens na tela consiste, em grande parte, em assegurar que tudo aconteça no momento correto. O problema reside na produção da varredura do feixe luminoso dentro do monitor. Ou seja, ela não é controlada pelo computador.

Quando não está conectado a nenhum micro, o monitor produz um feixe que varre a tela sessenta vezes por segundo, formando um campo iluminado regular. Para transformar esse campo em imagem é necessário "ligar" e "desligar" o feixe exatamente no mesmo lugar a cada 1/60 de segundo. Qualquer instabilidade nesse processo resultará numa tremulação irritante, que, no melhor dos casos, torna a tela cansativa para o olhar.

Todo o processo depende dos "pulsos de sincronização" e do brilho produzidos pelo computador e enviados ao monitor.

Há dois tipos de pulso de sincronização: um para cada linha da imagem e outro para cada imagem completa. No final de cada ciclo, um pulso curto informa ao monitor que a imagem está completa, obrigando o feixe de elétrons (e, com isso, o ponto que ele produz) a retornar ao canto superior esquerdo da tela para repetir o ciclo.

O processo é o mesmo ao final de cada linha. O monitor recebe um pulso informando que a linha se completou, e o feixe de elétrons volta ao lado esquerdo da tela para iniciar a linha seguinte.

Há vários tipos de monitor, enquadrados em duas categorias principais — coloridos e monocromáticos — subdivididas nos diferentes tipos de sinal que podem receber.

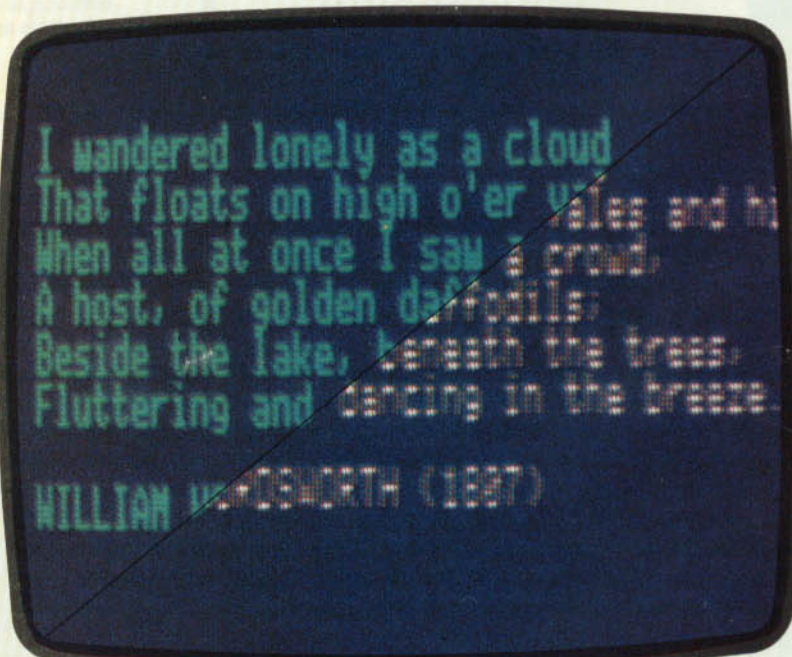
PIXEL

Palavra derivada de picture cell (célula da imagem), indica o menor conjunto de pontos da tela que pode ser acessado individualmente — ou seja: o bloco de construção das imagens.

Coloridos e monocromáticos

Os monitores monocromáticos mais utilizados são muito simples e trabalham com sinal composto: os pulsos de sincronização combinam-se com o nível de brilho num único sinal que o monitor utiliza para produzir a imagem.

Monitor x televisão

A tela mostra a grande diferença de qualidade de imagem obtida com monitor monocromático (no caso, o Monitor III, da Apple) e com aparelho de televisão comum, usados para apresentar a saída de um processador de texto.

Alguns monitores em cores funcionam com esquema semelhante, mas, devido à maior complexidade dos sinais, estão mais próximos dos aparelhos de televisão e operam em geral com os mesmos sistemas de codificação e decodificação.

Os tipos principais, PAL, SECAM e NTSC — nome dos sistemas usados na transmissão de televisão em diferentes partes do mundo —, representam métodos de codificar as cores básicas (vermelho, verde e azul) e os dois pulsos de sincronização.

Existe outro método, em que os vários sinais são enviados separadamente ao monitor. Embora haja variações sutis nos esquemas usados, estes são conhecidos como RGB, devido aos canhões de cores (Red, Green, Blue — vermelho, verde e azul). O mais simples é o TTL (transistor-transistor logic — lógica transistor-transistor), no qual as cores têm apenas dois estados: ligado e desligado. As oito cores usuais resultam das várias combinações dessas três.

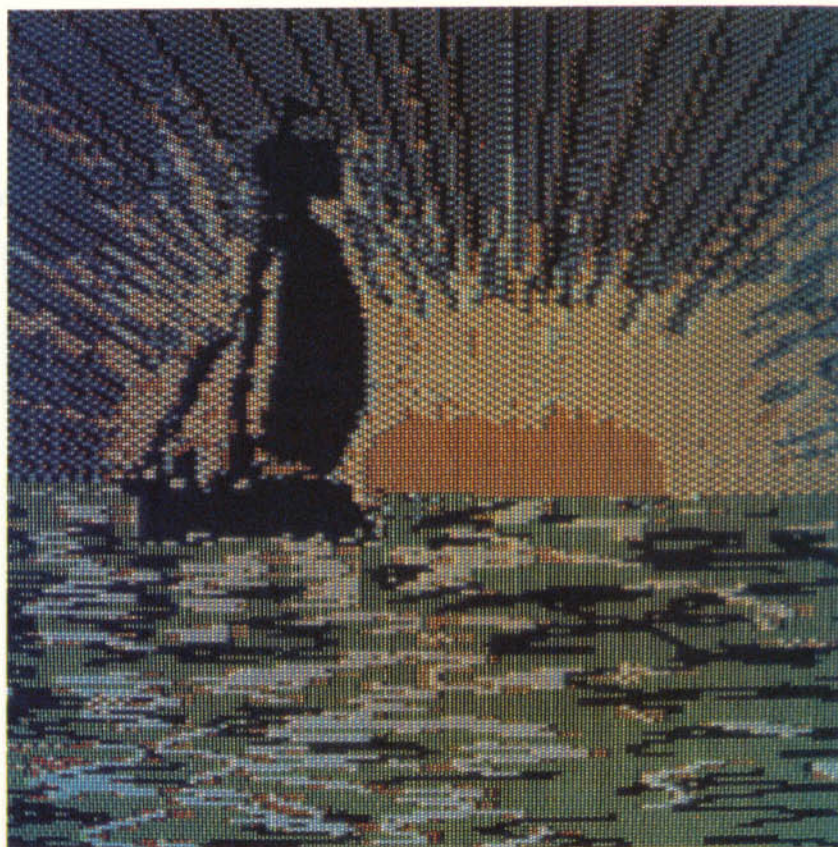
Produz-se maior variedade de cores pela intensificação de cada cor básica. Embora a intensificação seja gradual, o resultado é conhecido como monitor "analógico" ou "linear". A maioria dos monitores em cores é encontrada tanto na versão TTL como na analógica, com pouca diferença de preço.

O TM90PSN, de fabricação japonesa, é um monitor interessante e adaptável. Possui um mecanismo seletor cuja função é verificar o sinal que está entrando e passar o monitor para o modo apropriado. Assim, o TM90PSN recebe qualquer tipo de entrada, vantagem que lhe valerá a popularização, pois serve como vídeo universal — para computador, videoteipe, videodisco e outros equipamentos.

Resolução e largura de banda

Na escolha do monitor, um dos aspectos a considerar é a quantidade total de pontos que o circuito pode acessar — isso define a resolução do aparelho. Essa quantidade depende do tamanho do pixel. De modo geral, quanto menor o pixel, melhor a resolução.

Para calcular a resolução, determina-se em primeiro lugar o número máximo de caracteres na tela — número de caracteres em cada linha vezes o número de linhas. Multiplicando o total obtido pelas dimensões da matriz de caracteres chega-se ao número total de pontos. A matriz é o total de pontos que cabe no retângulo da tela ocupado por um caractere. Numa tela comum de 80 colunas por 24 linhas, por exemplo, o total de caracteres é 80 x 24. Se a matriz for de 7 x 9, isto é, um retângulo com sete pontos por nove, a resolução será dada por (80 x 7) x (24 x 9), ou 560 x 216. O número máximo de pontos será o resultado, 120.960. Em geral, esse número está entre 10.000 e 1,5 milhão.

Os 120.960 pontos do exemplo são varridos pelo feixe eletrônico sessenta vezes por segundo — que é a freqüência da rede. Assim, o circuito de controle deve ligar e desligar o feixe luminoso 7.257.600 vezes por segundo (120.960 x 60).

Combinação de três
Nos monitores coloridos, ideais para a produção de gráficos e figuras, as oito cores usuais resultam da combinação das três cores básicas.
A intensificação das cores básicas, no entanto, permite aumentar a variedade de cores.

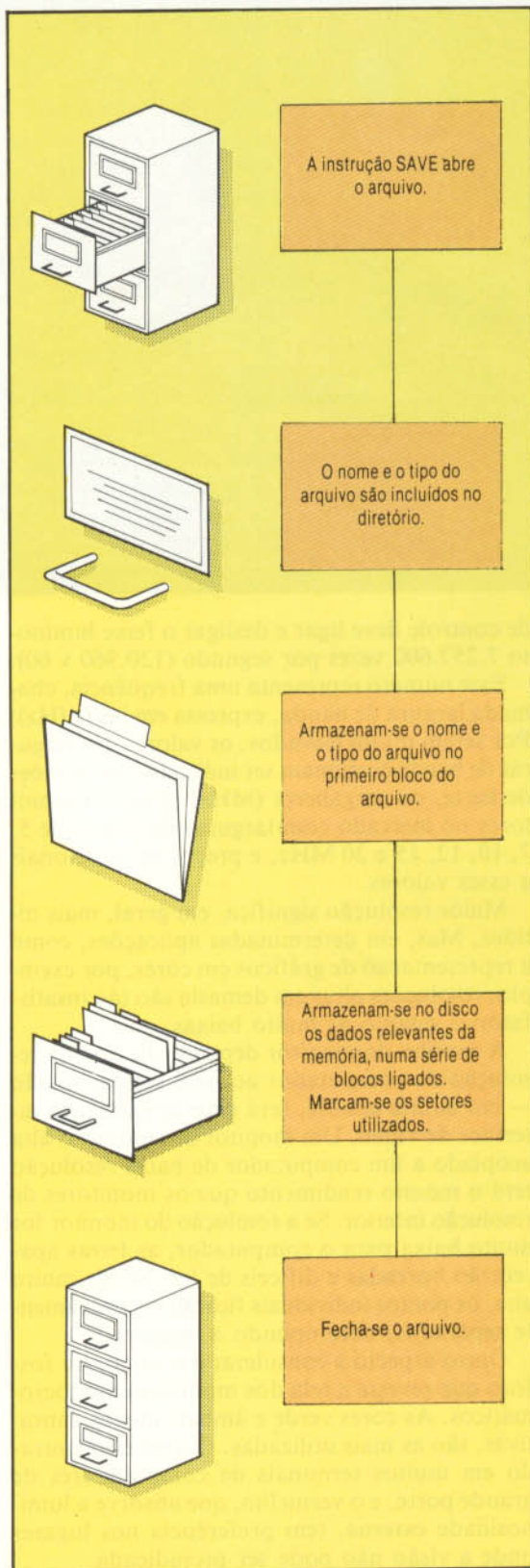
Esse número representa uma freqüência, chamada largura de banda, expressa em hertz (Hz). Por serem muito elevados, os valores das larguras de banda costumam ser indicados em milhões de hertz, ou megahertz (MHz). Existem monitores no mercado com larguras de banda de 5, 7, 10, 12, 15 e 20 MHz, e preços proporcionais a esses valores.

Maior resolução significa, em geral, mais nitidez. Mas, em determinadas aplicações, como a representação de gráficos em cores, por exemplo, resoluções altas em demasia são tão insatisfatórias quanto as muito baixas.

A escolha do monitor depende do tipo de resolução do computador ao qual será acoplado — em última análise, será determinada pela interface de vídeo. Um monitor de resolução alta acoplado a um computador de baixa resolução terá o mesmo rendimento que os monitores de resolução inferior. Se a resolução do monitor for muito baixa para o computador, as letras aparecerão borradas e difíceis de ler. Se for muito alta, os pontos individuais ficarão excessivamente separados, decompondo a imagem.

Outro aspecto a considerar é o da cor do fósforo que reveste a tela dos monitores monocromáticos. As cores verde e âmbar, menos cansativas, são as mais utilizadas. O azul é encontrado em muitos terminais de computadores de grande porte, e o vermelho, que absorve a luminosidade externa, tem preferência nos lugares onde a visão não pode ser prejudicada.

MEMÓRIA ORGANIZADA

Arquivamento direto
O arquivo binário copia parte da memória. Pode consistir num programa, numa imagem da tela etc. O arquivamento é direto. Depois de se entrar com o arquivo no diretório, os dados são gravados numa série de setores interligados. O DOS também mantém uma lista de setores usados, para evitar que sejam sobregravados por um novo arquivo.

Os arquivos — dados armazenados em computador — podem ser binários, seqüenciais ou aleatórios. E devem estruturar-se segundo o tipo, o volume e a velocidade das informações que concentram.

Arquivo é o termo mais adequado para designar o armazenamento de dados em computador. Sob vários aspectos, esse método se assemelha bastante ao de guardar documentos e revistas em pasta de arquivo.

O controle de nossas atividades diárias exige recursos como manter agenda de compromissos ou preencher o canhoto de cheque, fiscalizando o saldo bancário. Nessas situações, verifica-se a necessidade de guardar informações e de acessá-las com rapidez. Porém, à medida que aumenta o volume de informações manipuladas, torna-se mais complexo seu armazenamento.

Para gerenciar qualquer projeto ou empreendimento é básico ter em conta as constantes mudanças das informações em projetos que envolvem diferentes pessoas, lugares, objetos e circunstâncias. Muitos dos problemas enfrentados pelas empresas devem-se às dificuldades de controle e interpretação das informações. Portanto, a administração eficiente dos negócios exige a adoção de métodos simples de arquivamento, classificação e acesso aos dados.

Administradores competentes percebem a necessidade de estruturar o armazenamento segundo o tipo, o volume e a velocidade das informações sob seu controle. Esses princípios permanecem os mesmos quando se trata de sistemas de computação, projetados para manipular e armazenar — com precisão e em alta velocidade — grandes quantidades de dados.

No centro de tudo está o sistema operacional de disco (DOS, Disk Operating System) do computador, administrando as informações fornecidas pelo "gerente", isto é, seu programa. Assim, a eficiência dos sistemas de armazenamento depende do tipo de estruturação dos dados adotados pelo DOS. Também se deve levar em conta a forma de manipular esse sistema.

Os micros utilizam os métodos de manipulação de arquivo desenvolvidos, a princípio, para computadores de médio e grande porte: arquivos binários, seqüenciais e aleatórios.

Um bom exemplo de arquivo binário — simples cópia de parte da memória RAM disponível para o usuário — é o programa gravado.

Imagine a área de RAM como um bloco de anotações. Se você quiser guardar apontamen-

Diretório do disco

Na tela está um diretório de disco produzido pelo utilitário STAT, em CP/M, fornecendo muito mais informações do que a maioria dos diretórios.

```
Recs  Bytes    Ext Acc Filename
   0     0K      1 R/W B:ACNTLIST.DTA
  11     2K      1 R/W B:ANT
  10     2K      1 R/W B:ANT.BAK
  64     8K      1 R/W B:ASM.COM
  16     2K      1 R/O B:CODELIST.DTA
  16     2K      1 R/W B:AUTOST.COM
   1     1K      1 R/W B:COMPANY.DTA
   2     1K      1 R/W B:CONTROL.DTA
  34     5K      1 R/W B:COPY.COM
  40     5K      1 R/W B:DDT.COM
   4     1K      1 R/W B:DUMP.COM
 250    32K      2 R/W B:INSTALL.COM
   4     1K      1 R/W B:JUNK
  16     2K      1 R/W B:LOAD.COM
   6     1K      1 R/W B:MICROLIN
  40     5K      1 R/W B:ML.COM
  86    11K      1 R/W B:MOVCPM.COM
  58     8K      1 R/W B:PIP.COM
   2     1K      1 R/W B:SCREEN.ASM
   1     1K      1 R/W B:SCREEN.COM
   4     1K      1 R/W B:SCREEN.DOC
  42     6K      1 R/W B:STAT.COM
Bytes Remaining On B: 85K
```

Recs
O número de registros (**records**) no arquivo pode variar. Aqui, eles têm 128 Kbytes de comprimento.

Bytes
O tamanho do arquivo é dado em Kbytes.

Ext
A extensão é uma medida alternativa do espaço em disco ocupado pelo arquivo.

Acc
Acesso: o arquivo pode ser marcado para leitura e gravação (R/W); ou apenas para leitura (R/O).

Nome do arquivo
O nome completo do arquivo se inicia com o nome da unidade de disco (A: ou B:), seguido pelo nome do arquivo (COPY, por exemplo) e, por último, pela extensão do arquivo (.COM, por exemplo), que fornece informações sobre o tipo de arquivo.

tos importantes ou desenhos interessantes, arranque as páginas em que eles estão registrados e guarde-as em lugar de fácil acesso.

Os arquivos binários funcionam da mesma forma. Ao receber um comando SAVE, o DOS registra no disco o nome do arquivo (FILENAME), marcando-o de forma especial como arquivo binário. Em seguida, a área relevante da memória é copiada byte a byte no disco. Armazena-se o programa em blocos interligados — com marcadores no final de cada bloco indicando o início do bloco seguinte — até chegar ao final do programa ou dos dados. O último bloco termina com um marcador de fim de arquivo.

Assim, demos nome a uma das páginas de nosso bloco de anotações e a guardamos numa gaveta de arquivo, além de termos incluído esse nome na lista de conteúdo da gaveta.

Ao receber um comando RETURN ou ENTER, cada linha de um programa em BASIC é codificada, com as palavras-chave da linguagem BASIC transformadas em números de 1 byte pelo interpretador BASIC. Isso porque os números são mais fáceis de manipular e para acessá-los basta sua decodificação em forma de texto.

Como a imagem da RAM, o arquivo binário também armazena códigos ASCII e dados binários. Gravar arquivos ASCII é útil para armazenar, por exemplo, o conteúdo da tela. Com esse recurso, os dados mostrados na tela são gravados e facilmente carregados na mesma área da memória.

Alguns DOS e algumas versões de BASIC permitem o armazenamento de programas em BASIC codificados em ASCII. Assim, sob a forma de um arquivo de texto pode-se editar o programa não codificado por meio de programas processadores de texto.

Apesar de muito fáceis de usar e controlar, os arquivos binários são limitados por dois fatores. Em primeiro lugar, só se gravam as informações em bloco contínuo de dados, o que obriga a recuperá-las da mesma forma. Para serem acessados, os arquivos binários devem estar integralmente carregados na memória. Além disso, o tamanho máximo de um arquivo é limitado pelo total de RAM disponível para o usuário.

Outras matérias desta série examinam os arquivos seqüenciais, que podem ser, dentro dos limites de espaço disponível no disco, tão grandes quanto necessário. Analisam também os arquivos de acesso aleatório, que permitem ao sistema operacional empregar diferentes métodos de arquivamento dos dados.

SOMAS MÁGICAS

O quadrado mágico — intrigante quebra-cabeça matemático — é um candidato ideal a experimentos. Sua implementação no microcomputador envolve alguns problemas interessantes.

O quadrado mágico é formado por linhas e colunas de números inteiros positivos, não repetidos, dispostos de modo que a soma de qualquer linha ou coluna tenha o mesmo resultado.

O zero não é permitido. Monta-se um quadrado mágico com qualquer quantidade de linhas, desde que igual à de colunas. Uma das soluções para o mais simples deles (ou seja, de 3 x 3) é esta:

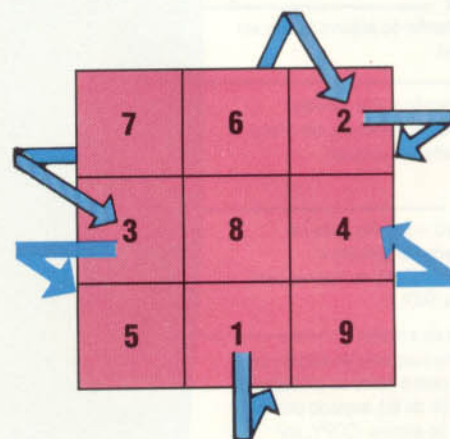
7	6	2	15
3	8	4	15
5	1	9	15
15	15	15	

Nesse exemplo, a soma de qualquer linha ou coluna é 15. Como exercício, tente construir um quadrado mágico de 5 x 5 usando números de 1 a 25. Embora aparente ser fácil, esse quebra-cabeça exige diversas tentativas até alcançar a solução.

Utilizando o microcomputador, a tarefa fica bem mais simples. Bastaria, por exemplo, escrever um programa que introduzisse os números no quadrado e testasse todas as linhas e colunas. Encontrar a solução por esse método, no entanto, pode levar horas, mesmo para quadrados pequenos. O ideal é antes descobrir um processo para gerar os números. O método exposto a seguir sempre funciona e pode poupá-lo desse trabalho.

Comece colocando o número 1 na célula do meio da linha inferior.

Para acrescentar o número seguinte (no caso, o 2), mova uma célula para baixo e uma para a direita. Se esse movimento o levar para fora do quadrado, vá para a primeira linha de cima, conservando a mesma coluna. Se o deslocamento para a direita sair dos limites, ocupe a primeira coluna da esquerda, mantendo a posição da linha.

Caso a célula seguinte já esteja ocupada, coloque o número uma posição à esquerda da última célula usada e continue o processo até preencher todo o quadrado, como na ilustração.

É fácil escrever um programa para gerar quadrados mágicos de qualquer tamanho. Começa-se por criar uma matriz quadrada vazia, do tamanho certo para o quadrado mágico desejado. Em seguida, recorre-se a um loop para preenchê-lo de acordo com as regras estabelecidas. Esse mé-

Quadrados de encantamento

Estes quadrados foram gerados e, depois, checados pelo programa. No quadrado de 15 x 15, observa-se uma seqüência de números de dois e três dígitos, que resulta do algoritmo usado em sua construção.

52	42	32	22	12	2	73	72	62
63	53	43	33	23	13	3	74	64
65	55	54	44	34	24	14	4	75
76	66	56	46	45	35	25	15	5
6	77	67	57	47	37	36	26	16
17	7	78	68	58	48	38	28	27
19	18	8	79	69	59	49	39	29
30	20	10	9	80	70	60	50	40
41	31	21	11	1	81	71	61	51

todo só funciona para quadrados mágicos com número ímpar de células (3 x 3, 5 x 5, 7 x 7, ...). Assim, o programa deve rejeitar matrizes com número par de células.

A clareza dos números na tela é o meio mais seguro de elaborar o quadrado. O alinhamento correto dos números em fileiras e colunas será simples se seu micro aceitar o comando PRINT USING. Se não, converta o número a ser impresso num string alfanumérico, usando caracteres de espaço, a fim de que todos os números tenham o mesmo comprimento. Para tanto, utilize a sub-rotina a seguir, que se adapta com facilidade a qualquer micro.

```
1000 REM Converte A em A$ e alinha
1010 A$ = STR(A)
1020 IF LEN (A$) < 3 THEN A$ = " " + A$: GOTO
1030 RETURN
```

O tamanho da tela costuma restringir a produção de grandes quadrados mágicos. As telas de 40 colunas comportam 13 colunas de dois dígitos mais as colunas de espaço, mas os quadrados de 13 x 13 incluem números de três algarismos. Nessas condições, o maior quadrado que cabe na tela é o de 9 x 9. Usando uma impressora, geram-se quadrados muito maiores. Em geral, as impressoras têm largura máxima de 132 colunas; quando se desejam quadrados maiores, podem-se imprimi-los em partes que serão unidas depois.

O objetivo principal desse projeto é criar o maior quadrado mágico compatível com a tela e apresentá-lo da forma mais clara. Faça seu programa e lembre-se de incluir comentários explicativos e uma rotina de verificação dos dados da entrada quanto ao tamanho pretendido.

130	114	98	82	66	50	34	18	2	211	210	194	178	162	146
147	131	115	99	83	67	51	35	19	3	212	196	195	179	163
164	148	132	116	100	84	68	52	36	20	4	213	197	181	180
166	165	149	133	117	101	85	69	53	37	21	5	214	198	182
183	167	151	150	134	118	102	86	70	54	38	22	6	215	199
200	184	168	152	136	135	119	103	87	71	55	39	23	7	216
217	201	185	169	153	137	121	120	104	88	72	56	40	24	8
9	218	202	186	170	154	138	122	106	105	89	73	57	41	25
26	10	219	203	187	171	155	139	123	107	91	90	74	58	42
43	27	11	220	204	188	172	156	140	124	108	92	76	75	59
60	44	28	12	221	205	189	173	157	141	125	109	93	77	61
62	46	45	29	13	222	206	190	174	158	142	126	110	94	78
79	63	47	31	30	14	223	207	191	175	159	143	127	111	95
96	80	64	48	32	16	15	224	208	192	176	160	144	128	112
113	97	81	65	49	33	17	1	225	209	193	177	161	145	129

```
10  REM **************************
20  REM *   QUADRADOS  MAGICOS   *
30  REM **************************
40  M=19:DIM A(M,M)
50  PRINT:PRINT "QUADRADOS MAGICOS"
60  PRINT:PRINT "QUANTAS LINHAS (1 A 19)";:
    INPUT S
70  IF S<0 OR S<>INT(S) THEN PRINT "ERRO":
    GOTO 60
80  IF S>M THEN PRINT "ERRO":GOTO 60
90  IF S/2=INT(S/2) THEN PRINT "ERRO, APENAS
    NUMERO IMPAR":GOTO 60
100 REM *** GERA QUADRADO ***
110 X=INT(S/2)+1:Y=S:C=1
120 A(X,Y)=C:C=C+1:IF C>S*S THEN GOTO 200
130 X=X+1:IF X>S THEN X=1
140 Y=Y+1:IF Y>S THEN Y=1
150 IF A(X,Y)<>0 THEN X=X-2:Y=Y-1
160 IF Y=0 THEN Y=S
170 IF X=0 THEN X=S
180 IF X=-1 THEN X=S-1
190 GOTO 120
200 REM *** MOSTRA QUADRADO ***
210 PRINT:PRINT
220 FOR Y=1 TO S:FOR X=1 TO S
230 A=A(X,Y):GOSUB 390   :PRINT A$;
240 NEXT X:PRINT:NEXT Y
250 REM *** VERIFICA LINHAS E COLUNAS ***
260 F=0
270 FOR Y=1 TO S:T=0
280 FOR X=1 TO S:T=T+A(X,Y):NEXT X
290 IF F=0 THEN U=T:F=1
300 IF T<>U THEN PRINT "ERRO - LINHA 1 E
    LINHA ";Y;" NAO COMBINAM":END
310 U=T:NEXT Y
320 FOR X=1 TO S:T=0
330 FOR Y=1 TO S:T=T+A(X,Y):NEXT Y
340 IF T<>U THEN PRINT "ERRO - LINHA 1 E
    COLUNA ";X;" NAO COMBINAM":END
350 U=T:NEXT X
360 PRINT "TODAS AS LINHAS E COLUNAS
    ADICIONADAS SOMAM ";T;
370 GET A$:IF A$="" THEN GOTO 370
380 END
390 REM *** CONVERTE NUMERO EM STRING ***
400 A$=STR$(A)
410 IF LEN(A$)<4 THEN A$=" "+A$:GOTO 410
420 RETURN
```

Variações

Embora escrito para a família Apple, este programa pode ser rodado em quase todos os micros. Para o CP 400 e o CP 500, troque apenas a instrução GET A$ da linha 370 por A$ = INKEY$. Devido à limitação da tela, é possível fazer quadrados de 7 x 7 no CP 400, 13 x 13 no CP 500, 9 x 9 no Apple de 40 colunas e 19 x 19 no Apple de 80 colunas.

Para a linha Sinclair, insira LET antes de todas as atribuições de valor. O programa pede o número de linhas (e, portanto, o de colunas) da matriz e verifica se é inteiro, positivo e ímpar. Daí, gera e exibe o quadrado mágico na tela. A partir da linha 250, checa os resultados. Se isso lhe parecer desnecessário, omita as linhas 250 e 360.

APPLE II*c*

Sem compatíveis no Brasil, o Apple II*c* (o *c* significa "compacto"), lançamento recente da família Apple, apresenta avanços em relação aos modelos II, II+ e II*e*, mas mantém compatibilidade com eles.

O Apple II*c* é um modelo portátil, com a metade da altura de seus antecessores, os Apple II, II+ e II*e*, e com uma unidade de disco embutida no lado direito. Pesa apenas 3,4 kg e já vem com uma alça embutida, que se transforma em apoio para colocar a máquina em ângulo confortável para o usuário.

Ao contrário das outras máquinas da família Apple, o II*c* é uma máquina com arquitetura fechada, sem slots (encaixes) para o acoplamento de expansões e periféricos. Em vez disso, já é dotado internamente da maioria das interfaces e das conexões que antes necessitariam de placas especiais. Há saídas para impressora, para um segundo drive, para modem, para TV doméstica, vídeo composto e RGB, e uma saída de nove pinos que serve para joystick ou mouse. O II*c* já vem com 128 Kbytes de memória RAM e 80 colunas de vídeo. Sua tela também foi consideravelmente melhorada. Além das duas opções no modo texto (40 ou 80 colunas), no modo gráfico ela apresenta dezesseis cores e três tipos de resolução: baixa, alta e superalta. No começo de 1985, as indústrias Apple planejavam o lançamento de um visor de cristal líquido (LCD) de 24 por 80 colunas. Com isso e com o sistema de alimentação a pilha também em estudo, o Apple II*c* se tornará totalmente portátil.

Conjunto de disquetes
O Apple II*c* vem com um conjunto de cinco disquetes, três dos quais com noções de BASIC, LOGO e instruções sobre a máquina. O quarto é um software integrado que combina planilha financeira e banco de dados. E, por fim, um disquete que contém o sistema operacional PRODOS.

Emprega o sistema operacional DOS 3.3 ou o novo PRODOS, que trabalha com armazenamento dos arquivos em árvore. Cada arquivo nesse sistema é incluído numa estrutura hierárquica, facilitando a organização e o acesso dos arquivos no disco. Inteiramente compatível com os outros Apple II, o modelo compacto dispõe de imensa quantidade de softwares: mais de 17.000 programas, de todos os tipos, foram escritos para o Apple II. Além destes, para o próprio II*c* desenvolveu-se o Appleworks, um software que combina processador de texto, banco de dados e planilha financeira. Janelas na tela facilitam a total integração das três aplicações. Dispõe também do MousePaint, um gerador de imagens que vem com o mouse opcional.

Acompanham o Apple II*c* cinco discos: um com os programas utilitários do sistema operacional PRODOS e quatro com demonstrações a respeito do próprio equipamento, o software Apple-works, o BASIC e o LOGO.

Saída padrão RS232 para impressora

Saída para vídeo composto

Fonte de alimentação
O II*c* tem uma fonte interna (bateria) de 12 volts e requer um transformador para a corrente de 110/220 VAC.

Saída para vídeo RGB
Um pequeno adaptador PAL conectado a esta saída permite que o II*c* se acople a uma TV comum.

Controladores de entrada e saída (I/O)
Estes chips controlam o teclado e todas as saídas para periféricos.

Saída de áudio
Conexão para amplificador de som externo.

ROM
Com 16 K, contém as rotinas de processamento interno e o BASIC Applesoft.

MousePaint
Este software gráfico acompanha o mouse, disponível para os modelos II+, II*e* e II*c*. É uma versão simplificada do MacPaint da Macintosh e permite criar desenhos na tela com muita facilidade. Um menu apresenta opções sob a forma de figuras. O MousePaint aproveita as cores e a alta resolução do II*c*. O mouse não exige interface especial para se acoplar ao Apple II*c*.

Unidade de disco interna
Para discos de 143 K, compatível com o Apple II+ e o Apple IIe.

Saída para disco
Uma unidade de disco, externa, conecta-se aqui, diretamente.

Saída para modem

Saída de nove pinos
Para joystick ou mouse.

RAM de 128 K
O dobro da RAM padrão do Apple IIe.

Microprocessador
O chip 65C02 é uma versão CMOS do chip 6502. Consome menos energia do que a versão 6502; portanto, pode ser operado a pilha.

APPLE II*c*

MICROPROCESSADOR
65C02

CLOCK
1 MHz

MEMÓRIA
16 Kbytes de ROM e 128 Kbytes de RAM

SISTEMA OPERACIONAL
DOS 3.3 ou PRODOS

VÍDEO
Monitor de fósforo verde opcional.
Modo texto: 24 linhas de 40 ou 80 caracteres.
Modo gráfico: três níveis de resolução: baixa (40 x 40 pixels), alta (240 x 192) e superalta (560 x 192), com dezesseis cores.

TECLADO
Profissional, com 63 teclas, sendo quatro de movimentação do cursor. Dispõe de uma chave para alternar entre 40 e 80 caracteres e de outra para transformar o teclado QWERTY ou DVORAK.

LINGUAGENS
Residentes: ASSEMBLER e BASIC (versão Applesoft). Disponíveis: LOGO, PASCAL, FORTRAN.

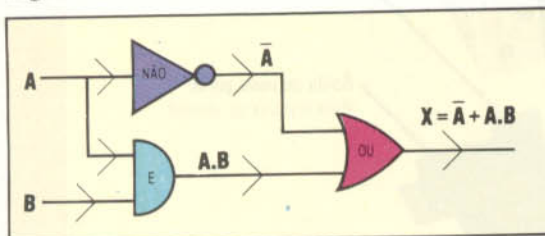
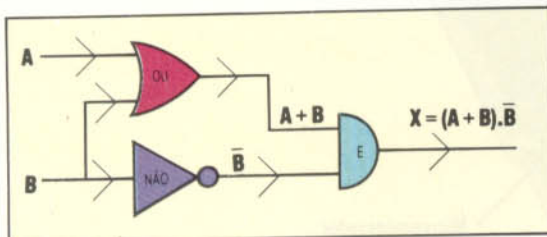
PERIFÉRICOS
Conexões internas para impressora, unidade de disco, modem, joystick, mouse, amplificador de som, TV doméstica, vídeo composto e RGB.

DOCUMENTAÇÃO
Ótimo guia do usuário, bem ilustrado, e um disco de demonstração interativa, *Apple presents Apple*.

A ESTRUTURA DOS BLOCOS DE ADIÇÃO

Com os três elementos básicos da lógica (E, OU e NÃO) produzem-se circuitos lógicos simples. Esta parte do curso investiga o modo pelo qual tais circuitos permitem executar a função da adição, considerando ainda a ordem da execução das operações.

O sistema que emprega a notação algébrica para descrever relações lógicas é a álgebra booleana. A esse sistema deve-se a simplificação matemática dos circuitos lógicos dos computadores, que permite executar uma função com número reduzido de portas lógicas. Em conseqüência, aumenta consideravelmente a velocidade de operação da máquina.

As expressões para a saída a partir das três portas lógicas (em notação booleana), E ($A \cdot B$), OU ($A + B$) e NÃO ($\bar{A}$), servem também para circuitos mais complexos. Por exemplo, a expressão booleana $X = \bar{A} + A \cdot B$ representa o seguinte circuito:

A ordem de execução das operações E e OU é relevante. A regra, nesse caso, é simples: E tem prioridade com relação a OU (bem como a NÃO). Se for preciso inverter a ordem de prioridade, usam-se parênteses, como no exemplo: $X = (A + B) \cdot \bar{B}$. Para essa expressão, operam-se as duas entradas inicialmente em OU e, depois, em E, com a negativa de B. Abaixo, o diagrama do circuito:

Ao desenhar um circuito lógico, convém que você comece pela saída e trabalhe para trás, em direção às entradas. Esse método sempre produz melhores esquemas de diagramas de circuitos.

OU: inclusivo e exclusivo

Há dois significados possíveis para a palavra OU na linguagem cotidiana:

Um OU outro OU ambos.

Um OU outro, mas não ambos.

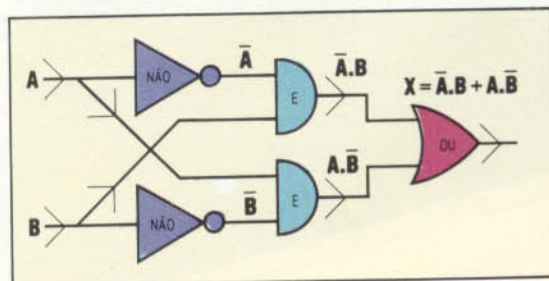
Por exemplo, numa competição de proprietários de veículos de duas rodas, podem participar os donos de motocicleta OU de bicicleta (ou ambos). Nesse caso, o OU é inclusivo. Por outro lado, uma pessoa pode ser alta OU baixa (mas não ambos). O uso da palavra OU, nesse exemplo, exclui a possibilidade de que as duas afirmações sejam verdadeiras.

Nos circuitos lógicos, essa operação do OU exclusivo (ou XOR) é um instrumento útil, que se forma com um conjunto de portas E, OU e NÃO. A tabela de validação para o XOR é:

ENTRADAS		SAÍDAS
A	B	$A \forall B$
0	0	0
0	1	1
1	0	1
1	1	0

Pela análise da segunda e da terceira linhas da tabela de validação, chega-se à conclusão de que a saída será 1 se:

NÃO (A) for operado em E com B
OU
A for operado em E com NÃO (B).

Essas duas possibilidades são representadas pela expressão booleana: $X = \bar{A} \cdot B + A \cdot \bar{B}$, da qual resulta um circuito de cinco portas capaz de produzir operação OU exclusivo:

Aplicando a álgebra booleana é possível simplificar esse circuito tornando-o de quatro portas.

Aritmética e portas lógicas

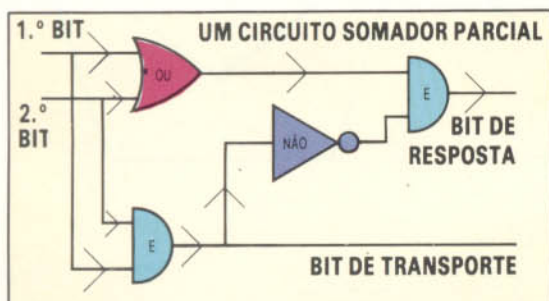
Embora a maioria dos micros execute toda a série de funções aritméticas, apenas a adição cabe aos circuitos lógicos. Realizam-se as demais, como subtração, multiplicação e divisão, por meio de uma combinação de software e de circuitos "somadores" do hardware para controlar o movimento das configurações de bits.

O conhecimento dos circuitos necessários para efetuar a adição binária exige que antes se examine o processo em si. Assim, considere, por exemplo, a adição binária:

	8s	4s	2s	1s
	0	1	0	1
	0	1	1	1
	1	1	0	0
TRANSPORTES	1	1	1	

Tomando uma coluna da soma isoladamente — por exemplo, a coluna dos dois — é possível relacionar suas várias entradas e saídas. *Entradas:* os dois bits a serem somados e o bit de transporte da coluna anterior. *Saídas:* o bit colocado na coluna dos dois como resposta e o bit a ser transportado para a coluna seguinte. O recebimento dessas entradas e a produção da saída correta cabem ao somador. O funcionamento desse dispositivo pode ser compreendido mediante uma versão simplificada, o somador parcial, que não considera a possibilidade de um transporte da coluna anterior. Isso reduz o problema a um circuito com duas entradas e duas saídas. E daí resulta uma tabela de validação para o circuito de um somador parcial:

ENTRADAS		SAÍDAS	
1.º BIT	2.º BIT	BIT DE TRANSPORTE	BIT DE RESPOSTA
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

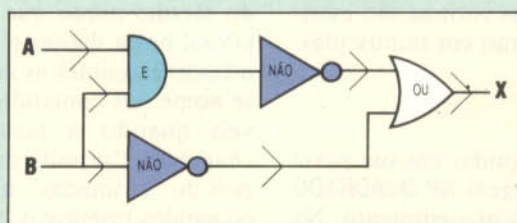
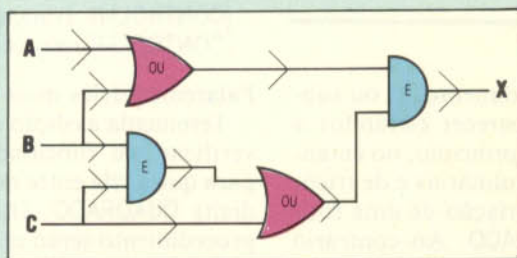
Pela tabela, percebe-se que o bit de transporte será 1 se o primeiro bit E o segundo forem ambos 1. O bit de resposta é formado pela operação em OU das entradas, desde que o bit de transporte não seja igual a 1. A saída do bit de resposta será 1 se o primeiro bit for 1 OU se o segundo for 1, e NÃO se o bit de transporte for 1. O circuito lógico abaixo produzirá as saídas desejadas:

EXERCÍCIO 2

1) Desenhe os circuitos lógicos para as expressões booleanas:

a) $X = (A + B)\cdot C$
b) $X = A\cdot B + (A + C)$
c) $X = \bar{A}\cdot B + (A + B)$
d) $X = \overline{A\cdot B}\cdot(A + B)$

2) Escreva a expressão booleana usando A e B como entradas para o bit de transporte e para o bit de resposta de um circuito somador parcial.

3) Escreva as expressões booleanas para os circuitos:

Respostas na próxima matéria desta seção.

Respostas do exercício anterior

1)

PASSOU NO EXAME DE HABILITAÇÃO	ACOMPANHADO POR UM MOTORISTA QUALIFICADO	PODE DIRIGIR UM CARRO
FALSO	FALSO	FALSO
FALSO	VERDADEIRO	VERDADEIRO
VERDADEIRO	FALSO	VERDADEIRO
VERDADEIRO	VERDADEIRO	VERDADEIRO

2)

GRAVADOR CASSETE DISPONÍVEL	UNIDADE DE DISCOS DISPONÍVEL	DESENVOLVIDO PARA COMPUTADOR DIFERENTE	O PROGRAMA SERÁ CARREGADO
FALSO	FALSO	FALSO	FALSO
FALSO	FALSO	VERDADEIRO	FALSO
FALSO	VERDADEIRO	FALSO	VERDADEIRO
FALSO	VERDADEIRO	VERDADEIRO	FALSO
VERDADEIRO	FALSO	FALSO	VERDADEIRO
VERDADEIRO	FALSO	VERDADEIRO	FALSO
VERDADEIRO	VERDADEIRO	FALSO	VERDADEIRO
VERDADEIRO	VERDADEIRO	VERDADEIRO	FALSO

3)

A	B	P	Q	C
0	0	1	0	0
0	1	1	1	1
1	0	0	1	0
1	1	0	1	0

A ARTE DA TARTARUGA

Na linguagem LOGO, além de operar no modo imediato, o programador pode definir como rotina uma seqüência de comandos sob nomes específicos. Para carregá-los, basta escrever o nome.

À primeira vista, os "procedimentos", ou sub-rotinas, do LOGO podem parecer estranhos a muitos programadores. Seu princípio, no entanto, é o mesmo das receitas culinárias e de tricô. Este capítulo começa pela criação de uma rotina LOGO, chamada QUADRADO. Ao contrário dos comandos — sempre digitados em letras maiúsculas —, os nomes das rotinas são escritos tanto em maiúsculas como em minúsculas.

De início digita-se:

```
EDITE QUADRADO
```

A tela ficará limpa. Em seguida, em sua parte superior aparecerá a mensagem AP QUADRADO — um lembrete do nome do procedimento. Na parte inferior da tela, surgirá a mensagem: EDICAO: CTRL-C = FIM, CTRL-G = CANCELAR. Um tanto enigmática, essa mensagem serve para ajudá-lo a operar em modo de edição. Isso significa apenas que o LOGO não está operando em modo imediato, mas esperando uma rotina. Após digitar a rotina, pressione as teclas [CONTROL]-C para registrá-la. Se for necessário interromper e começar de novo, pressione [CONTROL]-G.

Quando o LOGO estiver em modo EDITE, nada do que se digitar será executado. O LOGO apenas recebe e armazena a seqüência de instruções em seu "dicionário" sob o nome dado à rotina. Complete a definição do procedimento QUADRADO, introduzindo os comandos:

```
AP QUADRADO
REPITA 4 [FR 50 DI 90]
FIM
```

Para fazer com que o LOGO registre esse procedimento, pressione CONTROL-C. Na tela surgirá a mensagem QUADRADO OK. Se houver algum erro, interrompa o processo, pressionando o CONTROL-G. Nesse caso, refaz-se o procedimento desde o início, ou usam-se os comandos de edição em modo EDITE para introduzir as correções. Em modo imediato, o editor de linhas permite apenas correções na linha que acabou de ser digitada. Digitando EDITE, seguido de um espaço e o nome do procedimento (por exemplo, EDITE QUADRADO), e depois teclando RETURN, aparecerá no alto da tela o nome do procedimento com o cursor logo abaixo. Isso permite posicionar o cursor a fim de corrigir o que for necessário. Além de mover-se com as setas, o cursor obedece aos comandos:

```
[ESC] elimina o caractere à esquerda do cursor
[CONTROL]-D elimina o caractere sob o cursor
[CONTROL]-N avança uma linha
[CONTROL]-P retrocede uma linha
```

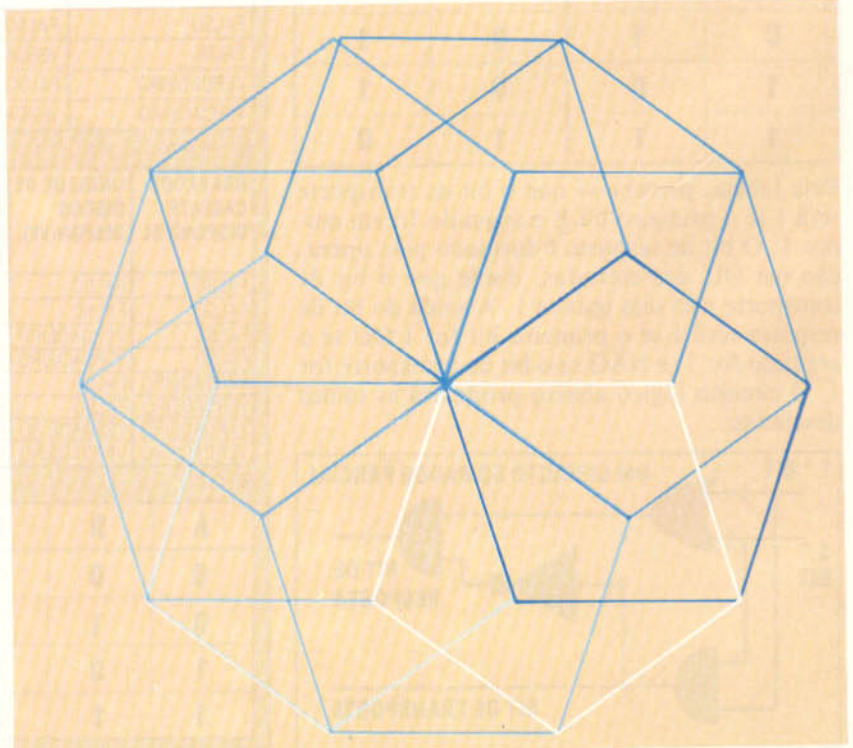
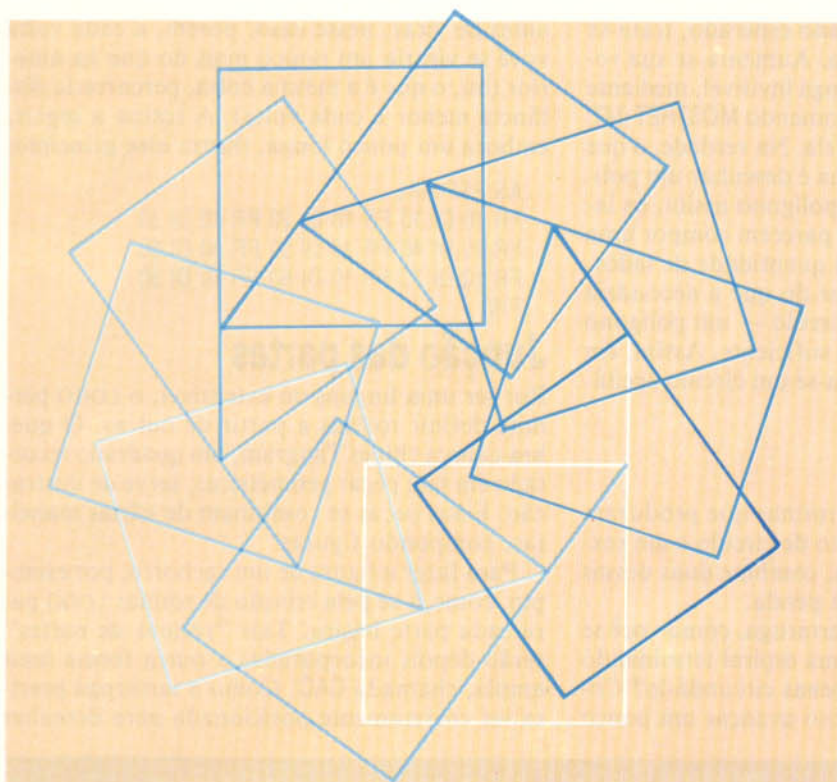
Falaremos deles mais adiante.

Terminada a edição e gravação da rotina, resta verificar seu funcionamento. Digite DESENHE para que a tela entre no modo gráfico. A seguir, digite QUADRADO. Os comandos definidos no procedimento serão executados e a tartaruga desenhará o quadrado. Usam-se os procedimentos do mesmo modo que os comandos básicos do LOGO: basta digitar o nome do procedimento e o LOGO executará as instruções definidas sob esse nome. Os comandos originais — já disponíveis quando o LOGO é carregado — são chamados "primitivos". As novas rotinas, depois de "ensinadas" no LOGO, são usadas como comandos primitivos. O LOGO é, assim, uma linguagem extensível, capaz de armazenar uma biblioteca de comandos que atenda a necessidades específicas.

Quando uma rotina não produz os resultados esperados, é simples modificá-la. A rotina do

Simplificação

EDITE	ED
SEMT	ST
MOSTRET	MT
LISTAR	LI

PENTÁGONOS

QUADRADOS

exemplo dado traça um quadrado cujos lados medem 50 unidades. Para obter um quadrado com lados de apenas 30 unidades, retorne ao modo EDITE, digitando:

```
ED QUADRADO
```

O texto da rotina aparece na tela. Utilizando o editor de tela, substitua 50 por 30. Em seguida, grave o novo procedimento, pressionando CONTROL-C. A seguir, digite QUADRADO e observe se o tamanho do quadrado é o desejado.

Como exercício, tente desenvolver procedimentos que produzam figuras complexas a partir das formas criadas no capítulo anterior: triângulos, retângulos, pentágonos, estrelas etc. Utilize o comando REPITA com esses procedimentos. Por exemplo, para o pentágono, defina o procedimento PENT como:

```
REPITA 5 [FR 50 DI 72]
```

A seguir, experimente:

```
REPITA 10 [PENT DI 36]
```

O comando REPITA permite traçar figuras complexas com rapidez e facilidade. Tente criar uma figura a partir de quadrados:

```
REPITA 10 [QUADRADO DI 36 FR 25]
```

As ilustrações mostram os resultados desses procedimentos.

Respostas do exercício 1

Triângulo:
REPITA 3 [FR 50 DI 120]
Pentágono:
REPITA 5 [FR 50 DI 72]
Hexágono:
REPITA 6 [FR 60 DI 60]
Retângulo:
REPITA 2 [FR 25 DI 90 FR 50 DI 90]
Estrela de cinco pontas:
REPITA 5 [FR 50 DI 144]
Estrela de dez pontas:
REPITA 10 [FR 50 DI 108]
Paralelogramo:
REPITA 2 [FR 25 DI 70 FR 50 DI 110]

Geometria cartesiana no LOGO

A geometria da tartaruga baseia-se nas propriedades intrínsecas das figuras. É o oposto do que ocorre na geometria cartesiana, em que as figuras são definidas em relação a um sistema externo de coordenadas ortogonais. Os movimentos da tartaruga são determinados pelo seu próprio avanço — segundo um número fixo de unidades — a partir de sua posição inicial na tela. Para desenhar um triângulo, por exemplo, a rotina correta é REPITA 3[FR 50 DI 120], e não REPITA 3[FR 50 DI 60], que resulta na metade de um hexágono. Isso porque o ângulo a ser considerado é o formado com a trajetória da tartaruga, isto é, 120°, em vez de 60°. Na geometria cartesiana, a tela é imaginada como uma rede quadriculada, com unidades dispostas em um eixo vertical e outro horizontal. Cada ponto da rede possui um valor numérico específico e os movimentos são definidos em relação a esses referenciais absolutos. Apesar disso, também é possível mover a tartaruga utilizando as coordenadas ortogonais.

Por exemplo, o comando DEFXY 20 30 move a tartaruga de sua posição inicial para o ponto (20,30). Acionando o comando COLORIDO (CL), a tartaruga traça uma linha à medida que se move. O comando SEMCOR (SC), ao contrário, faz com que a tartaruga se mova sem deixar rastro. A origem (0,0) fica no centro da tela.

As rotinas seguintes têm como resultado a mesma figura e ilustram a diferença entre a geometria da tartaruga e a geometria cartesiana:

```
AP QUADRADO1
REPITA 4 [FR 50 DI 90]
FIM
AP QUADRADO2
DEFXY 0 50
DEFXY 50 50
DEFXY 50 0
DEFXY 00
FIM
```

Ao digitar QUADRADO1 ou QUADRADO2, depois de retornar ao modo desenho (comando DESENHE), a mesma figura será traçada na tela. Mas o que aconteceria se você quisesse girar os dois quadrados em 30°? No primeiro caso, a instrução DI 30 QUADRADO1 será suficiente para se obter a rotação da figura. A segunda rotina, contudo, precisa ser reescrita (pois opera com coordenadas absolutas na tela), o que não é tarefa simples. Mas há ocasiões em que o emprego da geometria cartesiana no LOGO facilita muito certas tarefas. Nesses exemplos, você perceberá que, para traçar linhas, DEFXY é muito mais rápido do que FRENTE.

Outra característica da geometria da tartaruga é a sua "miopia". A tartaruga realiza apenas um movimento por vez. Ao traçar as figuras, ela cumpre uma série de pequenos "passos". Colocando-se no lugar da tartaruga é mais fácil perceber as condições necessárias para desenhar uma figura circular. Seria preciso mover-se um pouco para a frente e depois virar-se também um pouco, repetindo a seqüência muitas vezes. Na linguagem LOGO, isso se traduz por:

```
AP CIRCULO
REPITA 360 [FR 1 DI 1]
FIM
```

Embora produza o resultado esperado, trata-se de uma rotina muito lenta. Aumenta-se sua velocidade tornando a tartaruga invisível, mediante o comando SEMT (ST). O comando MOSTRET (MT) faz com que ela retorne à tela. Na verdade, o que se pretende com essa rotina é desenhar um polígono de 360 lados. Num polígono assim, os lados são tão pequenos que parecem compor uma linha curva contínua. Tal quantidade de lados, no entanto, é muito maior do que a necessária para dar a ilusão de um círculo — um polígono de 36 lados é mais do que suficiente. Assim, em muito menos tempo, traça-se um círculo aceitável com a rotina:

```
AP CIRCULO
FIM
```

Tente agora desenvolver rotinas que produzam um semicírculo, um quarto de círculo e um sexto de círculo. Em seguida, combine duas dessas figuras para formar uma pétala.

Voltando ao lugar da tartaruga, como você se moveria para desenhar uma espiral terminando num ponto, em vez de apenas circundá-lo? Como no círculo, seria preciso avançar um pouco antes de virar; nesse caso, porém, a cada volta você se viraria um pouco mais do que na anterior (ou, o que é a mesma coisa, percorreria distância menor a cada volta). A rotina a seguir, embora um pouco longa, ilustra esse princípio:

```
AS ESPIRAL
FR 10 DI 10 FR 10 DI 20 FR 10 DI 30
FR 10 DI 40 FR 10 DI 50 FR 10 DI 60
FR 10 DI 70 FR 10 DI 80 FR 10 DI 90
FIM
```

Junção das partes

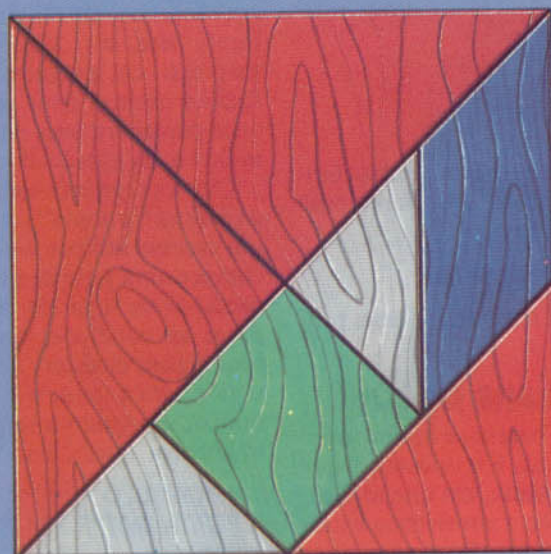
Por ser uma linguagem extensível, o LOGO permite definir rotinas a partir de outras. O quebra-cabeça chinês Tangram, um quadrado recortado em sete peças geométricas, serve de ilustração. Essas peças se combinam de várias maneiras, compondo figuras.

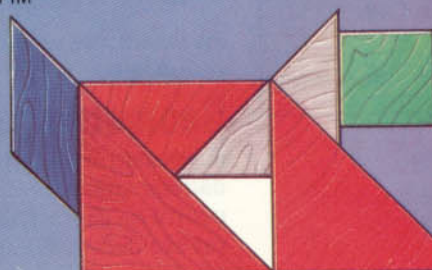
Para fazer a figura de um cachorro, por exemplo, começa-se pela criação de rotinas LOGO para cada parte básica. Tais "rotinas de partes" serão depois incorporadas a outra rotina mais ampla, chamada CAO. Como a tartaruga precisa ser corretamente posicionada para desenhar

Figuras simples

Ao lado encontram-se as rotinas LOGO para desenhar as sete peças básicas do quebra-cabeça Tangram. Há também um exemplo de rotina para a montagem da figura de um cachorro. Sob o nome CAO, essa rotina começa pelo traçado do triângulo que representa a perna traseira do animal.

Rotinas Tangram

```
AP QUADRADO
  REPITA 4 [FR 25 DI 90]
FIM
AP PAR
  REPITA 2 [FR 25 DI 45 FR 35 DI 135]
FIM
AP TRI1
  FR 25 DI 135 FR 35 DI 135 FR 25 DI 90
FIM
(Há dois triângulos desse tamanho.)
AP TRI2
  FR 35 DI 135 FR 50 DI 135 FR 35 DI 90
FIM
AP TRI3
  FR 50 DI 135 FR 71 DI 135 FR 50 DI 90
FIM
(Há dois triângulos desse tamanho.)
```

Programa do cão

```
AP CAO
  TRI3 MOVE1 PAR MOVE2 TRI2 MOVE3
  TRI1 MOVE4 TRI3 MOVE5 TRI1 MOVE6
  QUADRADO MOVE7
FIM
AP MOVE1
  SC FR 15 ES 45 CL
FIM
AP MOVE2
  SC DI 45 FR 35 ES 45 VO 35 CL
FIM
AP MOVE3
  SC ES 45 VO 25 CL
FIM
AP MOVE4
  SC DI 90 VO 25 CL
FIM
AP MOVE5
  SC FR 50 DI 45 CL
FIM
AP MOVE6
  SC FR 25 DI 135 FR 5 ES 90 CL
FIM
AP MOVE 6
  SC FR 25 DI 135 FR 5 ES 90 CL
FIM
AP MOVE7
  SC ES 90 FR 5 DI 45 VO 25 DI 45
  VO 50 ES 90 VO 50 CL
FIM
```

cada uma das peças, é necessário desenvolver também outras rotinas especiais — de MOVE1 até MOVE7.

Seria fácil criar esse desenho apenas interligando os comandos numa longa rotina. Entretanto, o método aqui aplicado é o da ordem "invertida": trata-se de dividir o problema em várias partes e solucioná-las uma por vez. Para o programador, o método é vantajoso, pois lhe permite criar uma rotina contendo sub-rotinas ainda não definidas. A rotina principal só pode ser rodada quando as sub-rotinas forem especificadas ou substituídas por pseudo-rotinas.

A rotina CAO, no exemplo, foi a primeira a ser escrita, seguida pelas sub-rotinas escritas separadamente e, por fim, pelas rotinas de posicionamento da tartaruga. Após desenvolver uma nova sub-rotina, rodava-se a rotina CAO para assegurar o encaixe perfeito de todos os seus componentes. Quando chegava a uma rotina MOVE ainda não escrita, o LOGO parava e enviava uma mensagem de erro. A partir do desenho, determinava-se com facilidade quando o erro estava na rotina principal ou na última rotina MOVE.

O exemplo demonstra que nem as sub-rotinas nem a rotina principal alteram o estado da tartaruga. Isso significa que ela retorna à posição em que estava antes da execução da rotina. Essas rotinas, chamadas "transparentes", facilitam a tarefa de encadear rotinas para elaboração de desenhos mais complexos. Considere a rotina CAO: sabendo que, depois de traçar uma das peças, a tartaruga retornará à posição inicial, é desnecessário conhecer o funcionamento interno das sub-rotinas para montar a rotina principal. E, com a rotina CAO transparente, ficou mais fácil incorporá-la em outra rotina — por exemplo, uma que multiplicasse a figura do cachorro na tela.

Área de trabalho LOGO

A memória operacional do LOGO consiste em uma lista de modos, cada um de cinco bytes. Depois de carregar o LOGO, ficam disponíveis entre 1.000 e 3.000 módulos, dependendo do modelo do micro. Os módulos são ocupados à medida que se gravam as rotinas. Outros módulos ainda podem ser usados enquanto as rotinas são executadas, ou no caso de se empregarem variáveis (que serão analisadas em outro artigo).

As rotinas gravadas constituem a área de trabalho. Faz-se a verificação das rotinas mantidas na área de trabalho mediante o comando LISTAR ROTINA (abreviado LI ROTINA). Para examinar uma rotina específica, use LISTAR (LI) — por exemplo, LI QUADRADO. Se a rotina não é mais necessária, esvazia-se o espaço de trabalho acionando o comando SEM. Por exemplo, SEM QUADRADO eliminará da memória a rotina QUADRADO. A eliminação de uma rotina libera o módulo por ela ocupado. O LOGO assinalará esses módulos livres, mas não os colocará de imediato na lista dos utilizáveis. Em lugar disso, continuará operando com os módulos anteriormente disponíveis, até que todos tenham sido usados. Só depois o LOGO pesquisará na memória os módulos liberados, formando com eles uma nova lista de módulos disponíveis. Esse processo é conhecido como "limpeza" e nele o LOGO parece hesitar por um ou dois segundos de vez em quando.

Exercício 2

A) Desenvolva rotinas que tracem as figuras do Tangram abaixo. É preciso antes descobrir como se dispõem as diferentes peças em cada figura.

B) Crie a rotina que desenha uma casa — basta um triângulo eqüilátero sobre um quadrado.

C) Escreva a rotina para um quadriculado com 5 x 5 quadrados.

D) Descreva a rotina usada para a estrela de seis pontas, de modo que seja formada de sub-rotinas.

Rotinas de armazenamento

Para registrar rotinas em disco de modo permanente, grava-se o espaço de trabalho sob a forma de arquivo. Supondo que DESENHOS seja o nome de um arquivo, digita-se GRAVE "DESENHOS (note que as aspas vêm apenas antes e não depois do nome-arquivo). O espaço de trabalho não será afetado por essa operação. O arquivo pode ser carregado por meio do comando LEIA "DESENHOS, que faz com que os procedimentos no arquivo sejam definidos e acrescentados à área de trabalho em uso. Quando se atribui a um procedimento o mesmo nome de outro já gravado, o novo procedimento substitui o anterior.

Outros comandos úteis para a manipulação de discos são VERDISCO e SEMARQUIVO. O primeiro fornece uma lista de todos os arquivos no disco. SEMARQUIVO "DESENHOS, por exemplo, elimina o arquivo DESENHOS do disco.

CUIDADOS COM O ESTILO

Documentar o programa é mais do que acrescentar comentários para o usuário. Um programa bem documentado contém informações suficientes para indicar seu objetivo e o modo de atingi-lo.

Observe, no quadro da página ao lado, a *Listagem 1* do programa apresentado em BASIC e PASCAL como exemplo. Ele se revela obscuro; dificilmente se entende o que ele faz. Percebe-se que aceita dois números pelo teclado, multiplica-os por outros dois, soma os resultados e imprime a resposta.

Sua tarefa específica só é revelada na segunda versão do programa *(Listagem 2)* — embora sem linhas de comentário, título ou documento externo.

Nomes (SANO e SMES) substituíram os números que na primeira nada significam. Números cujos valores não mudam enquanto o programa roda são chamados de "constantes". Algumas linguagens, como a PASCAL, têm notação especial para constantes. Na *Listagem 2,* as duas constantes são definidas separadamente das variáveis. Outras linguagens, como o BASIC, não fazem essa separação: as linhas 10 e 20 do programa em BASIC usam variáveis para definir as constantes. Dar nomes às constantes só interessa quando seu uso é freqüente; caso contrário, basta inserir no programa alguns comentários — linhas iniciadas por REM (de REMARK, ou seja, "comentário").

Uma segunda diferença ressalta: todos os nomes confusos das variáveis, representados apenas por uma letra, foram substituídos por palavras inteligíveis. ANOS (em vez de A), IDADESEG (em vez de E) e outras foram selecionadas por terem menos de dez caracteres e os dois primeiros serem diferentes entre si.

Em geral, constitui boa prática dar às variáveis nomes relacionados com o papel que desempenham no programa. Por exemplo: pode-se chamar de CONTADOR um contador de loops e atribuir seu primeiro e seu último valores a constantes ou a variáveis de valores predeterminados. Assim, um loop como FOR J = 1 TO 10-NEXT J representa-se por FOR CONTADOR = A TO M-NEXT CONTADOR.

Nomes longos de variáveis exigem mais digitação e memória, mas resultam em programas inteligíveis e facilmente debugáveis (depuráveis de seus erros).

Se a linguagem pela qual você optou usa apenas os dois primeiros caracteres de uma variável para identificá-la — o que ocorre com quase todas as versões do BASIC —, escolha nomes em que os dois primeiros caracteres diferem entre si. Isso porque, no caso, nomes longos de variáveis como CODNUM e COMP serão iguais para o computador.

Outra grande diferença entre as listagens-exemplo: a segunda usa mensagens bem explícitas ao pedir as entradas e ainda acrescenta uma explicação para cada saída (a linha PRINT, em BASIC, e a WRITE, em PASCAL).

Com isso, o programa torna-se mais inteligível para o usuário e para o próprio programador.

A diagramação do programa

Recursos simples, como deixar linhas em branco, alternar letras maiúsculas e minúsculas e alinhar as instruções por grupos, transformam uma impenetrável massa de símbolos em exposição lógica, organizada e legível.

Formatar um programa para a tela ou para a impressora é fundamental quando ele se vale de construções em loop (FOR-NEXT, WHILE-WEND, REPEAT-UNTIL) e sobretudo quando os loops contêm outros loops, pois a posição das instruções indica onde começa ou termina cada seqüência de instruções executada repetitivamente até que ocorra certa condição.

O BASIC permite pouca escolha quanto à apresentação do programa. Linguagens compiladas, como a PASCAL, são bem mais flexíveis. Nelas, escrevem-se os programas com o auxílio de um editor de texto. Já em BASIC — salvo na versão para o TRS-80 —, os recursos para edição mostram-se elementares.

Os comentários são o principal meio de documentação interna do programa. O BASIC vale-se de linhas de comentários: a palavra REM as inicia e o interpretador ignora tudo o que encontrar até o próximo marcador de fim de linha (: ou RETURN). Em outras linguagens — PASCAL, PL/1, PROLOG etc. —, os comentários ficam entre os sinais /* e */ ou { e }. Com esse sistema, os comentários podem ocupar mais de uma linha. Mas, se você esquecer o */, o resto do programa será considerado um comentário e ignorado pelo compilador.

Acrescente comentários sempre que uma explicação for necessária: quando começar programas ou novas sub-rotinas, definir constantes ou funções, escrever instruções complexas etc.

Por outro lado, evite comentários longos ou prolixos. Muitas vezes, basta um lembrete curto e direto, colocado antes de um bloco de instruções complexas. Coloque-o dentro da sub-rotina apenas quando sua presença não interferir na

compreensão da estrutura lógica do programa. A *Listagem 3* constitui um exemplo do bom uso de comentários.

Documentação externa

Manuais e especificações escritas são de produção difícil e tediosa. Pesquisas européias demonstraram que, em geral, os programadores só consultam documentação externa como último recurso. No entanto, quando bem usada, ela poupa esforços consideráveis.

As linguagens de quinta geração pretendem aumentar a produtividade do programador também por meio da criação automática de documentação, o que se pode conseguir pelo uso de informações selecionadas da fase de projeto do programa.

E uma das melhores maneiras de documentar seus programas, por enquanto, é usar o mesmo recurso. Mantenha sempre uma pasta para cada programa, enquanto você escreve. Guarde ali todas as anotações que fizer durante o projeto do programa — inclusive fluxogramas e esboços de algoritmos. É fundamental arquivar a vèrsão final do fluxograma usado para escrever as instruções.

Se você tem uma impressora, guarde também uma listagem do programa acabado. Em nossa versão completa do programa-exemplo, o primeiro comentário contém o nome e a data. Essa precaução permite, sempre que se modificar o programa, a atualização da data para se saber qual é a última versão.

Registre

Linha TRS-80

HEXADECIMAL PARA DECIMAL

Se você tem um CP 500 ou um CP 300 versão cassete, deve sentir falta de uma rotina para transformar números hexadecimais (de base 16) em decimais (de base 10). Endereços de memória e certos valores para POKE são normalmente fornecidos em notação hexadecimal.

Para transformá-los em decimais, você precisa ter uma calculadora à mão e realizar uma série de contas. Mas com este pequeno programa em BASIC entra-se com o número em hexadecimal e obtém-se imediatamente seu equivalente decimal.

```
10 INPUT "TECLE NUMERO HEXA: ";A$
20 P=4096:R=0:FOR X=1 TO 4:N=ASC(MID$
   (A$,X,1)):IF N<48 OR N>70 OR (N<65
   AND N>57) OR LEN (A$)<>4 THEN PRINT
   "ERRO":GOTO 10
30 B=N-48+7*(N>64):R=R+B*P:P=P/16:NEXT X
40 PRINT "DECIMAL: ";R:GOTO 10
```

Lembre-se de que a notação hexadecimal usa os dez dígitos numéricos e as letras de A até F. Entre com números de quatro dígitos de 0000 até FFFF para obter o resultado decimal de 0 até 65.535.

Documentação apropriada

Listagem 1

BASIC

```
(A) 10 INPUT A,B
    20 C=A*31536000
    30 D=B*2592000
    40 E=C+D
    50 PRINT E
```

PASCAL

```
(B) PROGRAM ABCDE (INPUT,OUTPUT);
    VAR A,B,C,D,E:INTEGER;
    BEGIN
    READ(A,B);
    C:=A*31536000;
    D:=B*2592000;
    E:=C+D;
    WRITELN(E);
    END.
```

Listagem 2

BASIC

```
(A) 10 AND=31536000
    20 MES=2592000
    30 PRINT "ENTRE SUA IDADE (ANOS DEPOIS MESES SEPARADOS
       POR UMA VIRGULA" ;
    40 INPUT NANOS,NMESES
    50 SANO=NANOS*ANO
    60 SMES=NESES*MES
    70 IDADESEGS=SANO+SMES
    80 PRINT "SUA IDADE EM SEGUNDOS E APROXIMADAMENTE
       ";IDADESEGS
```

PASCAL

```
(B) PROGRAM IDADESEGS (INPUT,OUTPUT);
    CONST
      AND=31536000;
      MES=2592000;
    VAR
      NANOS,NMESES,SANO,SMES,IDADESEGS:INTEGER;
    BEGIN
      WRITE('ENTRE SUA IDADE (ANOS DPOIS MESES SEPARADOS POR
      UMA VIRGULA) ');
      READ (NANOS,NMESES);
      SANO:=NANOS*ANO
      SMES:=NMESES*MES;
      IDADESEGS:=SANO+SMES
      WRITELN('SUA IDADE EM SEGUNDOS E IDADESEGS):
    END.
```

Listagem 3

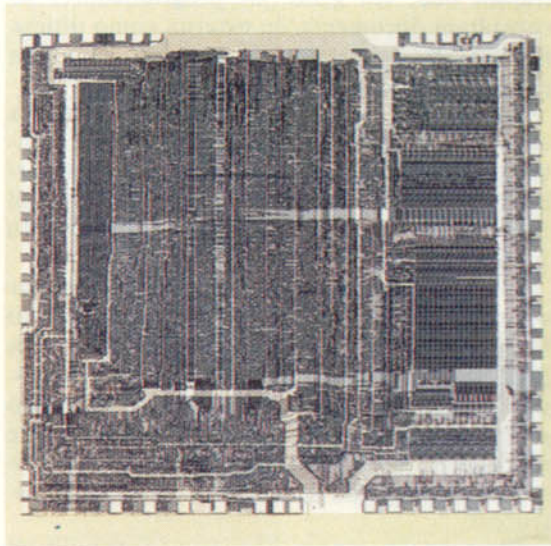
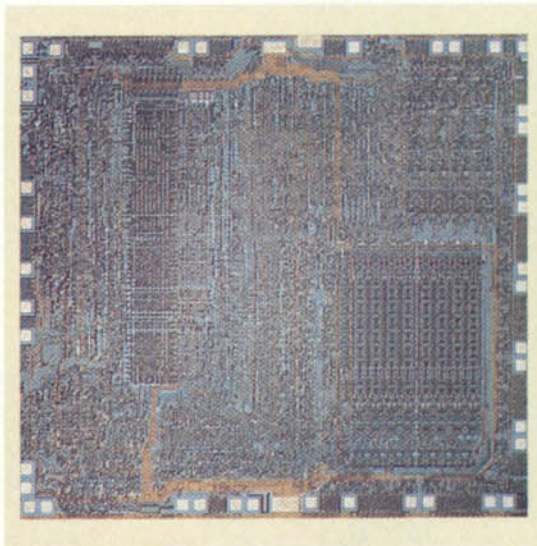
BASIC

```
(A) 10  REM "IDADE EM SEGUNDOS " MAIO DE 1985
    20  REM INPUT'S A IDADE EM ANOS E MESES (A,M) E
    30  REM USA UMA CONVERSAO APROXIMADA (1 MES= 30 DIAS)
    40  REM PARA RETORNAR A IDADE EM SEGUNDOS.
    50  REM
    60  AND=31536000:REM          SEGUNDOS EM 365 DIAS
    70  MES=2592000 :REM          SEGUNDOS EM 30  DIAS
    80  PRINT "ENTRE SUA IDADE (ANOS DEPOIS MESES SEPARADOS
        POR UMA VIRGULA ";
    90  INPUT NANOS,NMESES
    100 REM IDADE EM SEGUNDOS E' (IDADE EM ANOS * ANO EM
        SEGUNDO) MAIS (MESES DESDE ULTIMO ANIVERSARIO * MES
        EM SEGUNDO)
    110 SANO=NANOS*ANO
    120 SMES=NMESES*MES
    130 IDADESEGS=SANO+SMES
    140 PRINT "SUA IDADE EM SEGUNDOS E (APROXIMADAMENTE)
        ";IDADESEGS
```

PASCAL

```
(B) PROGRAM IDADESEGS (INPUT,OUTPUT);
    /*     MAIO DE 1985
        LE IDADE EM ANOS E MESES (ANO,MES) E USA UMA
        CONVERSAO APROXIMADA (MES= 30 DIAS) PARA
        RETORNAR IDADE EM SEGUNDOS.
    CONST
      AND=31536000;      /*SEGUNDOS EM 365 DIAS*/
      MES=2592000;       /*SEGUNDOS EM 30  DIAS*/
    VAR NANOS,NMESES,SANO,SMES,IDADESEGS:INTEGER;
    BEGIN
      WRITE('ENTRE SUA IDADE (MESES E ANOS SEPARADOS POR UMA
           VIRGULA) ');
      READ(NANOS,NMESES);
      /* IDADE EM SEGUNDOS E (IDADE EM ANOS * ANO EM
      GUNDOS) MAIS (MESES DESDE O ULTIMO ANIVERSARIO *MES EM
      SEGUNDOS)      */
      SANO:=NANOS*ANO;
      SMES:=NMESES*MES;
      IDADESEGS:=SANO+SMES;
      WRITELN('SUA IDADE EM SEGUNDOS E ',IDADESEGS);
    END.
```

OS DISSIDENTES

Conjunto de chips
Microfotografias que ilustram a complexidade dos circuitos do microprocessador. À esquerda, uma foto altamente ampliada do bem-sucedido Z80 da Zilog; à direita, um chip recente da série Z8000. O Z8000, um chip de 16 bits, possui desenho mais denso e mais conexões em torno das bordas.

Em 1977, a Zilog lançou o Z80. Em poucos anos, esse microprocessador e o 6502 tornaram possível o que antes parecia ficção científica: um computador em cada domicílio.

A história da empresa americana Zilog Incorporated começa no início da década de 70, quando Frederico Faggin e Masatoshi Shima, funcionários da Intel (fabricante de microchips), saíram dessa companhia. Anteriormente, os dois haviam participado do desenvolvimento do microchip 8080A (considerado o primeiro "computador num chip"). Com essa experiência, começaram a trabalhar em novos desenvolvimentos do microprocessador.

Frank de Weeger
Presidente da Zilog

O 8080 já era muito popular entre amadores e projetistas de computação; assim, Faggin e Shima decidiram desenhar o novo chip de modo que fosse compatível com o 8080, aproveitando deste a grande quantidade de software. Eles ampliaram o conjunto de instruções (a lista de comandos em código de máquina presente no microprocessador) pela introdução de registros extras, códigos de operação de 2 bytes e outras técnicas. Seu microprocessador — o Z80 — constitui um aperfeiçoamento considerável.

Essa revolução no hardware coincidiu com mudanças no software. Em 1972, Gary Kildall e John Torode escreveram o programa Control Program/Monitor (CP/M), que possibilitava a um microprocessador manipular o recém-lançado disco flexível.

Como Kildall era consultor da Intel, o programa foi projetado para rodar no 8080 e no 8055. O CP/M tornava-se rapidamente o sistema de manipulação de discos dominante para os microcomputadores. Com seu potente microprocessador compatível com o 8080, a Zilog estava estrategicamente situada e podia tomar a dianteira na corrida dos softwares para o CP/M.

Programação difícil

O caminho da Zilog não foi tão suave nos anos subseqüentes. Embora o Z80 ainda vendesse bem em meados da década de 80 — quando a companhia produzia cerca de 1 milhão de unidades por mês —, as tentativas de adaptar o chip para o mercado de 16 bits não obtiveram a reação esperada.

A primeira tentativa da Zilog com um processador de 16 bits foi o Z8000. Embora reconhecido como dispositivo muito potente, com amplo conjunto de instruções e grande número de registros, o chip revelou-se extremamente difícil de se programar.

O Z8000 encontrou ainda vários outros obstáculos. Era compatível com o Z80000 (ou Z80K), de 32 bits, mas não com o Z80. Não podia, portanto, aproveitar a amplitude e a variedade de programas escritos para o Z80 nos anos precedentes. Os fabricantes interessados nos chips de 16 bits para seus equipamentos voltaram-se, pouco a pouco, para outros microchips, com menos exigências.

Com o Z8000 pouco popular no mercado de micros, a Zilog voltou à mesa de projetos a fim de estudar o processador Z800 de 16 bits, compatível com o Z80. Paralelamente, a empresa de computadores Commodore anunciou que sua nova série de máquinas para uso comercial seria suprida com o Z8000.

DESCOBRINDO A SENHA

A maior parte dos casos de acesso ilegal aos computadores de grande porte por meio de microcomputadores e modems envolve órgãos governamentais e corporações multinacionais.

O filme *Jogos de guerra (War games)* fascinou a imaginação de muitos usuários de microcomputador. O personagem principal, usando um micro e um modem, consegue acesso ilegal a vários computadores, altera resultados dos exames de sua faculdade, reserva bilhetes aéreos e copia os jogos mais atuais. Mas as conseqüências da brincadeira tornam-se dramáticas quando ele acessa o computador do NORAD, órgão responsável pela defesa aérea dos Estados Unidos, e quase inicia uma guerra nuclear.

Distribuição ilegal

Embora negue o incidente, a Pepsi-Cola foi vítima de pirataria: alguém dos Estados Unidos obteve acesso ilegal ao computador dessa companhia americana no Canadá. Os piratas enviaram grandes carregamentc~ da bebida a várias localidades, transferindo quantias muito altas de dinheiro para suas contas.

Boa parte das violações de computador tem como autores adolescentes que usam micros domésticos e modems. Qualquer computador que permita acesso por telefone torna-se vulnerável. Em 1983, a realidade chegou perto da ficção, quando se suspeitou que dois "piratas" haviam acessado a rede de computadores do NORAD em Omaha, Nebrasca, nos EUA.

Os envolvidos eram dois adolescentes de Los Angeles, que violaram a Arpanet — rede secreta de computadores do Departamento de Defesa dos EUA. Valendo-se de um VIC-20 da Commodore e de um TRS-80 da Tandy, eles conseguiram dados de vários computadores da Arpanet. Esses equipamentos, em geral, pertencem a empresas contratadas pelo Departamento de Defesa, órgãos de pesquisa e universidades. Como o sistema compartilha dados científicos, nenhuma informação confidencial foi obtida. A facilidade com que os adolescentes violaram a Arpanet, porém, provocou um enorme constrangimento no Departamento de Defesa.

A facilidade de acesso, em muitos casos, não se deve a falhas do computador, mas à preguiça humana. Os usuários da Arpanet talvez não tivessem senhas muito criativas. Os meninos partiram da suposição correta de que a Universidade de Berkeley fosse usuária da Arpanet e entraram na rede com a óbvia senha UCB (Universidade da Califórnia em Berkeley). Com isso, ficaram livres para acessar qualquer computador da Arpanet, inclusive a instalação subterrânea central do NORAD, em Omaha.

As conseqüências da "invasão" apenas não foram mais sérias porque o escritório central do NORAD está ligado à Arpanet. Todos os computadores responsáveis pela defesa aérea do país foram instalados sob as montanhas Cheyne, no Colorado, e não se conectam às linhas públicas de telefone.

Mas há vários computadores que, ao contrário do sistema do NORAD, estão sujeitos a violações de piratas. Em julho de 1983, um grupo de adolescentes de Milwaukee acessou mais de sessenta computadores de universidades e corporações, além do Laboratório Nacional de Los Alamos, destinado à produção de armas. Mais uma vez, segundo as autoridades, nenhuma informação confidencial foi obtida. Os meninos vasculharam apenas registros, relatórios de rotina e mensagens. Acionou-se o FBI (Federal Bureau of Investigations) para descobrir como o grupo, autodenominado "414", realizou a façanha. O grupo defendeu-se afirmando que não havia medidas de segurança em nenhum dos computadores para os quais eles tinham telefonado.

Muitos outros casos não se tornam públicos porque as organizações evitam admitir que seus sofisticados computadores foram invadidos por garotos com micros. Muitas vezes, as próprias organizações desconhecem uma infiltração, dada a dificuldade de se detectar um usuário não autorizado — embora piratas mais insolentes deixem mensagens do tipo "vocês não me pegam" e se assinem "Destruidor de Sistemas" ou "Capitão Gancho".

Como ocorre a pirataria? Um pirata em potencial necessita apenas de um micro, um modem e engenhosidade. O primeiro obstáculo consiste em descobrir o número de telefone do computador. No caso de redes de acesso público — como a Telecom Gold, na Inglaterra, ou The Source, nos EUA —, isso não é difícil, já que os números são de domínio público. A pirataria se complica quando se trata de compu-

tadores particulares. Porém, dispondo-se pelo menos de uma vaga idéia da localização do computador, a técnica que o protagonista de *Jogos de guerra* empregou pode ser útil. Ele programou seu micro para discar todos os números possíveis em sua cidade. Se algum computador respondesse com um som característico, o micro registrava o número. Mas, se uma pessoa atendesse à chamada, o modem desligava e passava para o número seguinte. Para obter resultados mais rápidos, o jovem "herói" do filme usou um modem de discagem automática.

Uma vez realizada a conexão, haverá uma senha a responder. Certas redes permitem acesso limitado quando se digita "VISITANTE" ou "NOVO USUÁRIO" ou apenas se pressiona a tecla [RETURN]. No entanto, o verdadeiro pirata procura descobrir a senha por tentativa e erro. O que não é difícil, devido à falta de imaginação dos usuários, que utilizam como senha seu próprio nome ou palavras óbvias — "SEGREDO" e "SENHA", por exemplo. Quando recorrem a algarismos, a preferência recai em seqüências fáceis de lembrar, como a data de nascimento. Muitos computadores permitem várias tentativas antes de desconectar o intruso. E, mesmo assim, sem levantar suspeitas, pode-se ligar de novo para o computador.

Uma vez dentro do sistema, a maior parte dos piratas apenas examina os registros, vasculha os programas de jogos e "conversa" com outros piratas que estão na linha. Mas há os destruidores, que apagam registros, deixam mensagens obscenas e tentam bloquear todo o sistema. Uma invasão de sistema pode trazer conseqüências desastrosas para os usuários legítimos.

Enfrentar a máquina

Mesmo nos computadores de grande porte mais sofisticados, os programadores sempre deixam entradas de emergência no sistema para — sem passar pelas medidas de segurança — penetrar com rapidez no programa. Muitas vezes, esse recurso é desconhecido até pelos operadores do sistema.

A maior parte dos incidentes ocorre com computadores de universidades, pois eles permitem discagem externa e operam com acesso aberto. Contando com milhares de usuários, o acesso aberto é o meio mais prático para a operação desses sistemas. Só que, assim, ficam muito mais vulneráveis à pirataria.

Após entrar no sistema, os intrusos podem pular de um computador a outro, passando por usuários legítimos. Um estudante do campus da Universidade da Califórnia em San Jose encontrou uma brecha no projeto Talk, que permite aos estudantes conversarem, via computador e modem, com outros campi da Universidade da Califórnia. Vencendo o obstáculo local, o rapaz conseguiu comunicar-se com computadores de todo o país e mesmo do exterior.

Os motivos da pirataria, em geral, se resumem à emoção de vencer o sistema. Muitos piratas têm

Tentativa e erro
O método usado pelos piratas envolve várias tentativas, até que, em raras ocasiões, se obtém entrada não autorizada no sistema. Ainda que consiga localizar o computador, para validar sua entrada no sistema, o pirata depara com o seguinte diálogo:

```
CONNECT QX001001 14.32
    12/7/84
>
USER?
>HELP
USER?
>£$OFF
USER?
>UK001
PASSWORD?
>GUEST
PASSWORD?
>NEWUSER
PASSWORD?
>QWERTY
LOGOFF 15.13 CONNECT
    TIME = 0.13 MINS

CONNECT BYF990
    15.14.02 12/07/84
>
USER?
>UK001
PASSWORD?
>SYSOP
LOGON 15.15.07 12/07/84
HOST: BYF990/SPYLOM
USER: UK001
SERIAL NO: ZA180-7
PRIORITY: SUPERUSER
STATUS: ACTIVE
YOU ARE SYSOP
7 USER(S)
APP01 APP02 BYF7 BTY04
    BZX88 BZX02 SYSOP
>REMOVE ARP01
USERS(S) ARP01
    DISCONNECTED
>WHO IS BTY04
BTY04     CAREY DIMMIT,
    BTY LOPP, 742
    SILICON DV
HERTS 07662-093164
```

Pressionando [RETURN]: usuário, identifique-se
Nenhuma ajuda: computador não amigável
Primeira tentativa de identificação válida
Entrada invalidada
Segunda tentativa de identificação
Identificação aceita: pede a senha
Primeira tentativa
Recusa: segundo pedido
Segunda tentativa
Recusa: terceiro pedido
Tentativa desesperada

Depois da terceira tentativa fracassada de achar a senha, desconecta-se o usuário.

Pressione [RETURN]
Identificação válida: pede a senha
Primeira tentativa operador de sistema
Número de série do programa
Arquivos liberados
Usuário desconectado se não usar o sistema de maneira correta
Confirmação
Liste todos os usuários
Comando reservado ao operador de sistema
Usuário válido desconectado

seu código de ética particular e não danificam o sistema, divertindo-se apenas com a descoberta dos códigos. Esquecem, no entanto, que usar o tempo do computador sem pagar também é ilegítimo.

Os bancos sempre foram muito atingidos pelos crimes por computador, envolvendo a transferência de dinheiro para contas falsas. A dificuldade do cálculo exato dos prejuízos causados pela pirataria deve-se ao desinteresse das companhias em admitir publicamente que foram vítimas de fraudes por computador. O aumento no número de usuários de microcomputador e de redes de computadores que usam linhas telefônicas agravou ainda mais os problemas desse gênero. Os métodos de fraude são os mesmos, não importando tratar-se de adolescentes travessos, apagando registros e roubando tempo do computador, ou criminosos desviando dinheiro para as próprias contas.

Acerto ocasional
Algumas vezes, depois de várias tentativas ou por pura sorte, o pirata descobre a senha válida e entra no sistema. O diálogo fictício acima descreve como o usuário poderia obter informações confidenciais sobre um usuário legítimo.

CÓPIAS QUENTES

Para as atividades comerciais, as impressoras matriciais ou tipo margarida são as mais indicadas, por sua boa qualidade de impressão. Os usuários particulares contam com alternativas mais baratas.

Todo usuário precisou, em algum momento, de cópias impressas de texto ou de listagem de programas — e constatou o alto custo de uma boa impressão. As impressoras matriciais chegam a ser duas ou três vezes mais caras do que o próprio computador. E a compra de uma impressora tipo margarida não se justifica para a maioria dos usuários.

As impressoras térmicas, porém, representam uma alternativa de baixo custo. Para impressão, tais equipamentos dispensam o impacto de agulhas ou letras metálicas sobre a fita. A ação do calor grava os dados no papel. Para tanto, é indispensável o uso de papel termossensível.

Esse tipo de impressão requer uma matriz de elementos que, quando requisitados, se aquecem, fazendo a pequena área correspondente do papel mudar de cor.

A base do processo das impressoras matriciais está no impacto provocado pela matriz de agulhas, suportada por uma cabeça, que se move horizontalmente pela página, imprimindo os dados de forma barulhenta. Nesse sentido, as impressoras térmicas são mais vantajosas, dada a sua operação silenciosa. A Apple adotou a nova tecnologia e desenvolveu uma impressora térmica para seu computador, batizando-a de Silentype ("tipo silencioso").

As impressoras eletrostáticas representam um tipo intermediário entre as matriciais e as térmicas: não são tão barulhentas como as primeiras nem tão silenciosas quanto as térmicas. Tiveram larga aceitação no exterior, quando as indústrias Sinclair utilizaram o sistema em sua impressora ZX, vendida em grandes quantidades como periférico dos micros ZX-81 e Spectrum-ZX.

A cabeça de impressão das eletrostáticas compõe-se de um fio que se move sobre o papel recoberto de uma camada metálica. A cada caractere acionado, o circuito da impressora gera uma faísca que queima a camada metálica, revelando o papel preto do fundo. A Sinclair aperfeiçoou o sistema ao empregar duas cabeças ou dois fios para impressão. Mas ainda assim precisa de oito movimentos da cabeça para formar cada linha de caracteres. Como a impressora ZX imprime apenas 32 caracteres por linha, a velocidade de impressão é aceitável.

Impressora térmica Tandy
Com mais facilidade de adaptação do que as outras impressoras de baixo custo, a TP-10 da Tandy é compatível com toda a linha de microcomputadores dessa empresa.

A maior desvantagem das impressoras térmicas e eletrostáticas é o seu papel especial. Além de caros, tais papéis só estão disponíveis em rolos, o que dificulta o armazenamento.

A impressora térmica exige cuidados especiais na compra do papel, que deve ser adquirido no grau correto de sensibilidade. Caso contrário, a imagem não será revelada a contento, desbotando com o tempo ou com a exposição ao calor.

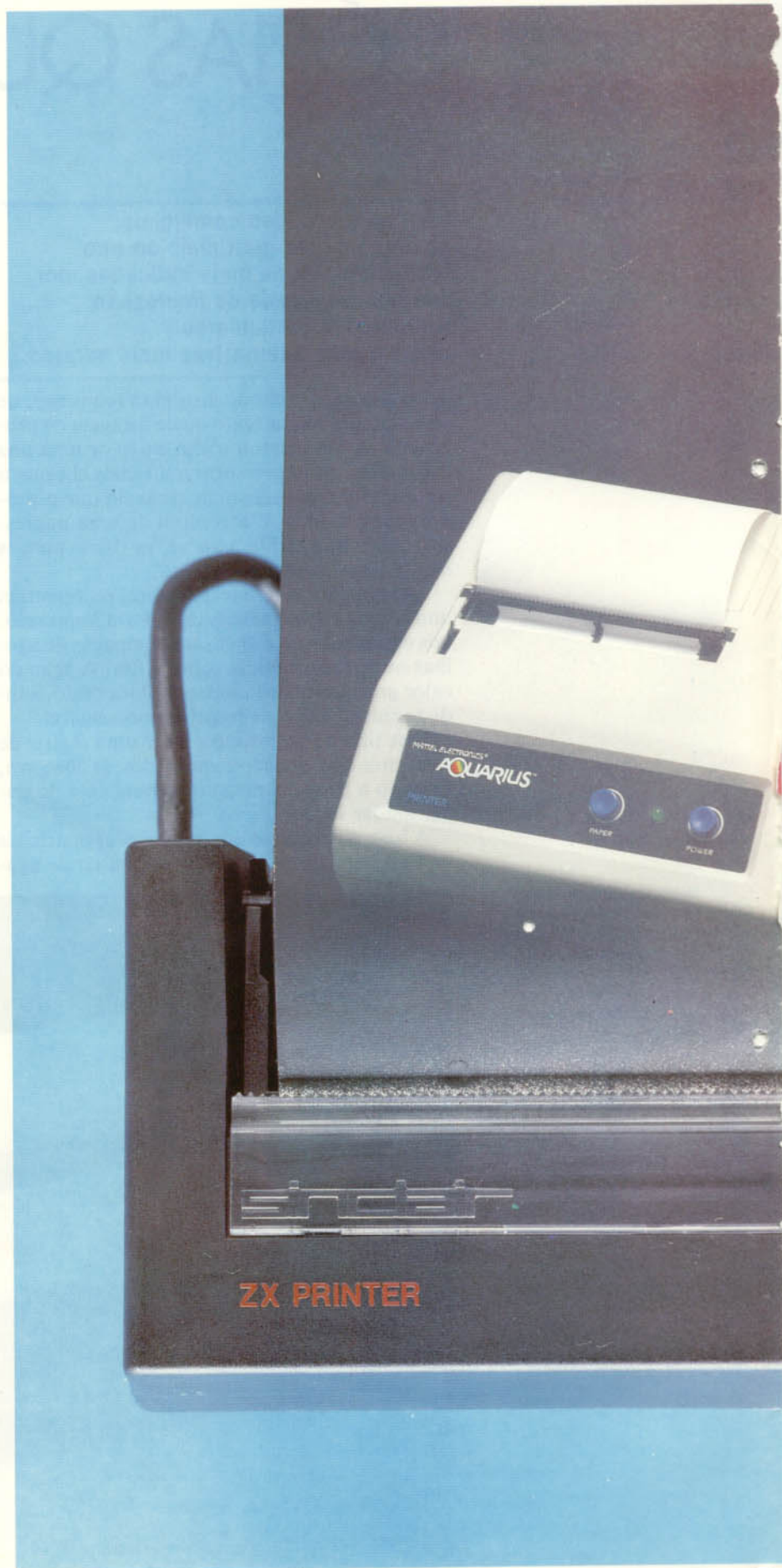
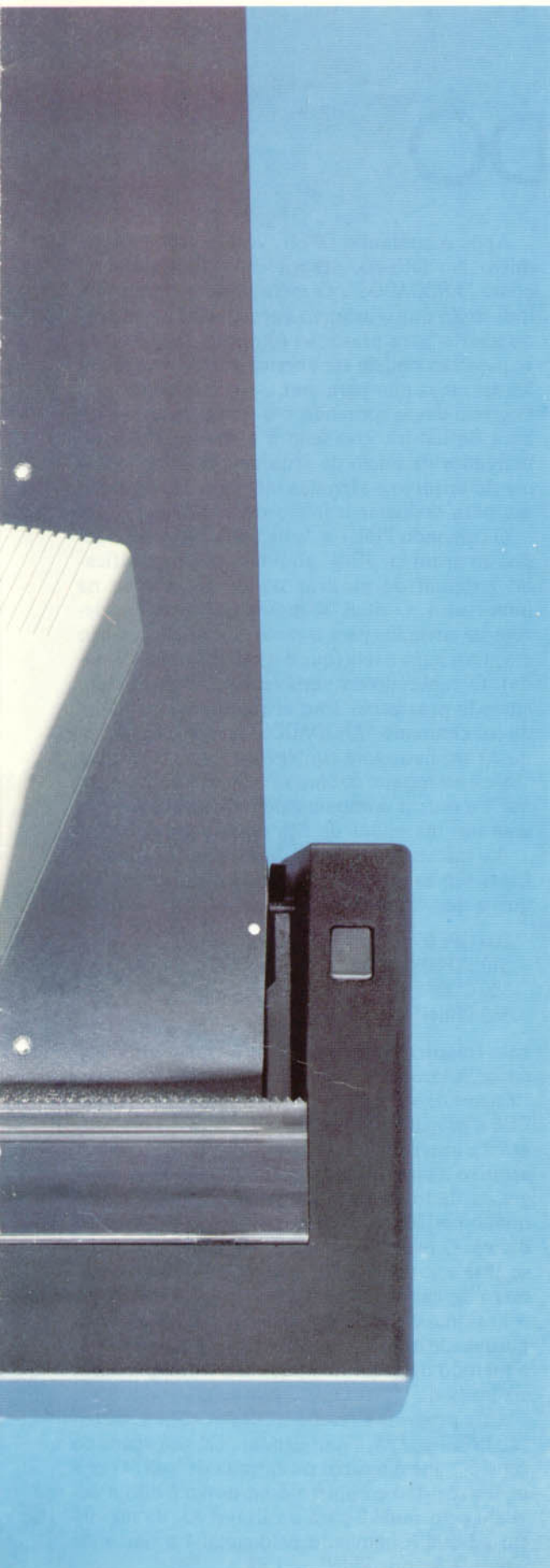
O papel eletrostático é ainda mais delicado e, se mãos úmidas o pegarem, dissolve-se a cobertura, borrando ou apagando a imagem. Nos dois casos, o meio mais indicado de garantir a durabilidade da imagem é tirar fotocópias.

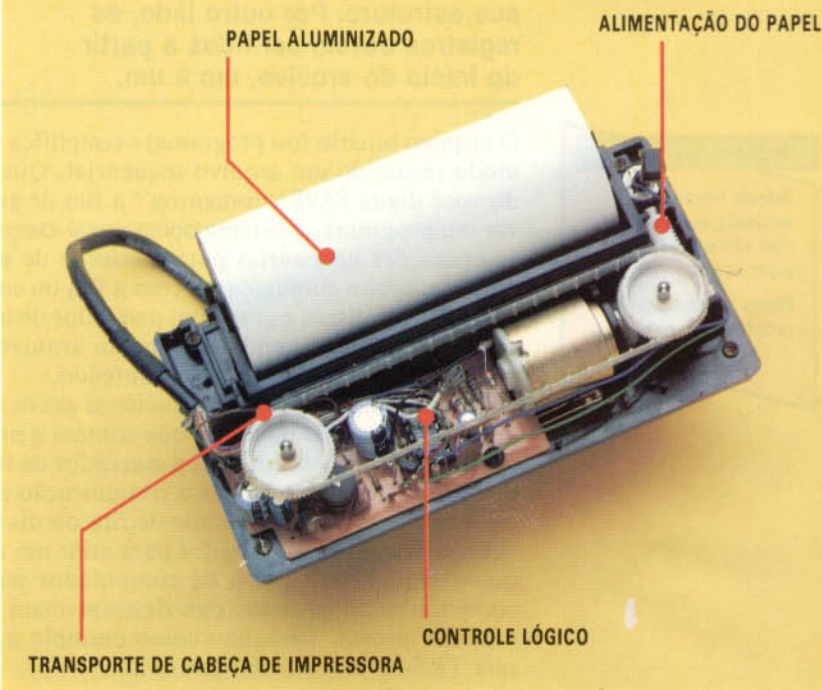
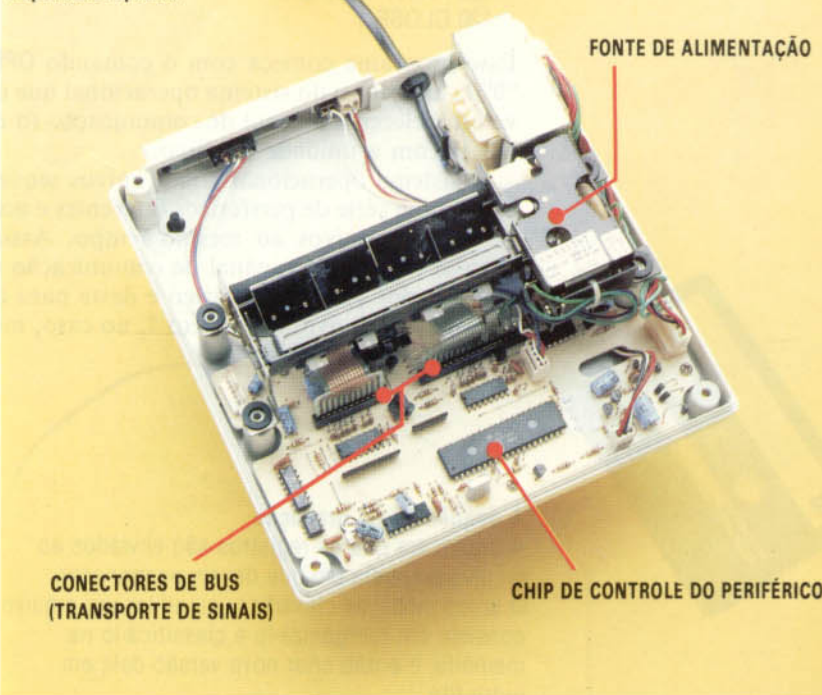
Apesar dos inconvenientes, esses dois tipos de impressora são a melhor alternativa para os usuários de microcomputador. Elas permitem que as cópias sejam feitas diretamente da tela, produzindo textos e gráficos, embora a qualidade de impressão não seja perfeita.

Por outro lado, os plotters têm-se revelado a mais nova alternativa para sistemas de impressão. Com uma ou mais canetas, eles desenham os caracteres como se usassem canetas esferográficas. Os plotters também são indicados na impressão de gráficos.

Sua grande vantagem está na utilização de papel comum. Além disso, seu custo não supera o das impressoras térmicas e eletrostáticas.

Apesar destas novas técnicas de impressão, os sistemas sem impacto continuam, com seu mecanismo simples e limitado, a ser a opção mais barata para o usuário de microcomputadores.

Big Brother
Além de mais barata, a impressora térmica Big Brother EP-22, de fabricação japonesa, incorpora três funções: trabalha como impressora térmica e como máquina de escrever portátil projetada para operar com papel térmico, além de permitir a impressão normal de impacto com fita. A EP-22 armazena em sua memória, mantida com bateria, até 2.000 caracteres, conservando os textos gravados mesmo quando desligada. O conteúdo da memória, no entanto, não pode ser totalmente modificado. Só é permitida a correção dos últimos dezesseis caracteres inseridos, pois o texto, antes de ser impresso, aparece para o usuário num visor de cristal líquido (LCD). A qualidade da impressão também não se compara à de uma impressora tipo margarida. As letras minúsculas, por exemplo, são cortadas nas extensões inferiores. Por outro lado, sendo uma impressora portátil para acompanhar microcomputadores, a EP-22 executa tarefas difíceis de realizar em qualquer tipo de impressora de impacto.

Impressora ZX
Introduzida como periférico do micro Sinclair ZX-81, a impressora ZX também é compatível com o Spectrum-ZX. Comercializada como uma alternativa econômica para a impressão de listagens de programas e textos, a ZX apresenta a desvantagem de funcionar apenas com papel eletrostático.

Impressora Aquarius
Apesar de dispor de econômica impressora de quatro cores, que usa canetas esferográficas, a Aquarius adotou a impressora térmica como opção de baixo custo para os usuários de seus microcomputadores. Mas, assim como a ZX, a Aquarius depende de papel especial.

ARQUIVO ORGANIZADO

Arquivos seqüenciais armazenam tudo o que uma variável contém. Não há restrições quanto a seu tamanho ou sua estrutura. Por outro lado, os registros devem ser lidos a partir do início do arquivo, um a um.

O arquivo binário (ou programa) exemplifica de modo resumido um arquivo seqüencial. Quando você digita SAVE "nomeprog" a fim de gravar seu programa, o sistema operacional executa as operações necessárias para a criação de um arquivo: abre a comunicação com a fita ou com a unidade de discos e grava um marcador de início de arquivo, contendo o nome do arquivo e algumas informações sobre o conteúdo.

Em seguida, o sistema operacional grava no arquivo o bloco de memória que contém o programa atual; depois, grava um marcador de fim de arquivo; e afinal encerra a comunicação entre o computador e a unidade de fita ou disco.

Os comandos BASIC usados para criar um arquivo seqüencial variam de computador para computador, mas todos eles desempenham as mesmas tarefas. Tomando como exemplo a linha TRS-80 modelo III, tem-se:

```
100 R$ = "Este é um registro do arquivo"
110 OPEN "0",1,"ARQDADOS"
120 PRINT # 1,R$
130 CLOSE 1
```

Esse programa começa com o comando OPEN "0",1, indicativo do sistema operacional que deve estabelecer um canal de comunicação (o canal 1) com a unidade de disco.

O sistema operacional cria arquivos seqüenciais numa série de periféricos diferentes e acessa vários arquivos ao mesmo tempo. Assim, deve-se especificar o canal de comunicação do computador para o periférico e deste para determinado arquivo. O número 1, no caso, indica a unidade de disco.

OPÇÃO E SEQÜÊNCIA

Default: Opção pré-assumida, que prevalece caso não se faça outra especificação.

String: Seqüência de caracteres alfanuméricos.

A seqüência da gravação
A ordem em que os registros são enviados ao arquivo é aquela em que devem permanecer. O único modo de classificar ou editar um arquivo consiste em reorganizá-lo e classificá-lo na memória, e então criar nova versão dele em outra fita.

Após o comando OPEN, vem o nome do arquivo. No TRS-80, este consiste em um nome tal como "ARQDADOS". O método de acesso é "0", indicando que o arquivo é seqüencial e está sendo aberto para gravação (output). Os arquivos seqüenciais podem ser abertos para gravação ou leitura, mas não para ambas ao mesmo tempo. O efeito desse comando é a energização da cabeça de leitura/gravação e a gravação de um marcador de início de arquivo que inclui o nome do arquivo e algumas informações do sistema para assinalar o início do arquivo.

O comando PRINT # 1, na linha 120, envia dados ao arquivo. PRINT aqui não tem o significado habitual de mostrar dados na tela ou na impressora. O sinal # indica que os dados devem ser enviados para o canal especificado, e não enviados para a tela (que é o canal default de saída). O conteúdo da variável R$ é, desse modo, enviado pelo canal 1 ao arquivo seqüencial em disco, chamado "ARQDADOS", onde será gravado. O arquivo agora contém um registro, o string "Este é um registro do arquivo". Close 1 fecha o canal 1 a outras comunicações e grava o arquivo com um marcador de fim de arquivo.

As informações serão lidas posteriormente. Estas são as instruções em BASIC para ler o arquivo que criamos:

```
500 OPEN "1",2,"ARQDADOS"
510 INPUT # 2,A$
520 CLOSE 2
530 PRINT A$
```

Este fragmento de programa, assim como o anterior, usa OPEN. Aqui este comando significa: "Encontre o início do arquivo chamado ARQDADOS e prepare-se para ler seus dados". No programa anterior, ele se traduzia por: "Crie um arquivo chamado ARQDADOS e prepare-se para gravar dados nele". O número de canal indica qual o periférico em uso (no caso, a unidade de disco). O número em si é irrelevante; basta que se abra e se feche um arquivo com o mesmo número de canal.

O nome e o tipo do arquivo são os mesmos porque identificam o arquivo a ser acessado, mas o método de acesso difere — "I" (input, leitura) ao invés de "0" (output, gravação).

Esse programa recebe os dados com a instrução INPUT # 2,A$, significando: "Entrada a partir do canal 2", que é o canal de entrada default. O primeiro registro completo no arquivo é lido e enviado pelo canal 2 para a variável A$, da mesma forma que foi enviado pelo canal 1 a partir da variável R$ por PRINT # 1,R$.

Arquive e encontre

```
10  REM ********* PARA LINHA TRS-80 **********
20  REM *      ARQUIVOS  SEQUENCIAIS         *
30  REM **************************************
40  CLS
50  FOR K=65 TO 90
60  Z$=CHR$(K):C$="GRAVANDO "+CHR$(K)
70  GOSUB 240
80  NEXT K
90  REM **************************************
100 REM *           LER  ARQUIVOS            *
110 REM **************************************
120 FOR L=0 TO 1 STEP 0:FOR M=1 TO 1
130 CLS:PRINT D$;"ACESSO AOS REGISTROS"
140 INPUT "PROCURAR STRING (* = FIM)";N$
150 L$=LEFT$(N$,1):IF L$="*" THEN GOTO 490
160 IF L$<"A" OR L$>"Z" THEN M=0
170 NEXT M
180 Z$=L$
190 GOSUB 370
200 PRINT TAB(5)"REGISTRO ";N;" (USE QQ TECLA)
210 A$=INKEY$:IF A$="" THEN GOTO 210
220 NEXT L
230 END
240 REM **************************************
250 REM *       GRAVAR UM ARQUIVO            *
260 REM **************************************
270 PRINT C$:INPUT "QUANTOS REGISTROS";R
280 OPEN "O",1,Z$
290 IF R=0 THEN PRINT#1,"*":CLOSE 1:RETURN
300 FOR I=1 TO R
310 PRINT C$:PRINT "REGISTRO #";I
320 INPUT "TEXTO...";R$
330 PRINT#1,R$
340 NEXT I
350 PRINT#1,"*":CLOSE 1
360 RETURN
370 REM **************************************
380 REM *           LER UM ARQUIVO           *
390 REM **************************************
400 PRINT "PROCURANDO ";L$;" DE ";N$
410 OPEN "I",1,Z$
420 FOR I=1 TO 100000
430 INPUT#1,R$
440 PRINT R$
450 IF R$="*" THEN N=0:I=100000
460 IF R$=N$ THEN N=I:I=100000
470 NEXT I:CLOSE 1:N$=""
480 RETURN
490 REM **************************************
500 REM *         FIM DO PROGRAMA            *
510 REM **************************************
520 PRINT "FIM DO PROGRAMA":END
```

50-80: Criar arquivos "A" a "Z".

120-180: Entrar com o registro a procurar, encontrar a letra inicial, identificar o arquivo que a contém.

190: Pesquisar o registro dentro do arquivo.

200-210: Mostrar o registro encontrado.

280: Abrir um arquivo para gravação.

290: Se o arquivo estiver vazio, gravar "*"; fechar o arquivo.

320-330: Entrar o texto do registro; gravá-lo no arquivo.

350: Gravar "*" como último registro; fechar o arquivo.

410: Abrir o arquivo para leitura.

450: Teste de último registro; interromper a pesquisa.

460: Verificar se o registro foi encontrado, interromper a pesquisa.

470: Fechar o arquivo.

Esse programa demonstra o uso de arquivos seqüenciais em disco, criando um índice simples, em ordem alfabética, que abrange 26 arquivos, um para cada letra do alfabeto. Em cada arquivo, você pode digitar registros começando com a letra correspondente. O arquivo pesquisado aparecerá na tela até que o registro seja encontrado; se não for encontrado, a mensagem "REGISTRO" aparecerá na tela.
O programa usa o sistema operacional em disco para obter acesso direto ao arquivo que contém o registro pesquisado; no entanto, o arquivo em si é lido seqüencialmente. Isso reduz o tempo de pesquisa.

Variações

O programa que apresentamos como exemplo trabalha com arquivos em disco, impróprios para a linha Sinclair.

Para trabalhar com arquivos seqüenciais em micros da linha Apple, o processo é um pouco diferente: usa-se o caractere de controle CHR$(4) (que equivale a teclar [CONTROL] D) e as instruções READ e WRITE, além de OPEN e CLOSE.

Assim, o comando para abrir o arquivo seria:

PRINT CHR$(4);"OPEN NOMEARQ,S6,D1"

Nessa instrução, S6 significa o slot de número 6, e D1 a unidade de disco 1.

Para fechar o mesmo arquivo, usa-se esta instrução:

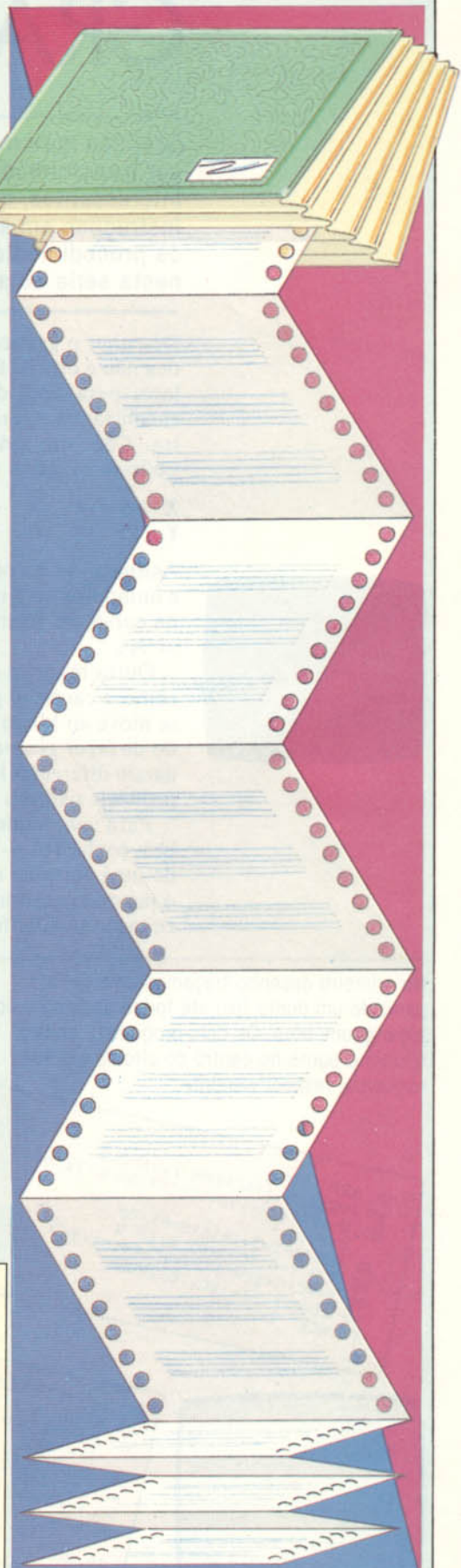
PRINT CHR$(4);"CLOSE NOMEARQ"

É dispensável mencionar o slot e o número do drive, na instrução.

Para ler um arquivo, necessitamos destas duas instruções:

PRINT CHR$(4);"READ NOMEARQ,REGISTRO"
INPUT R$

Para gravar dados no arquivo, as instruções são:

PRINT CHR$(4);"WRITE NOMEARQ,REGISTRO"
PRINT R$

GRÁFICOS

Gráficos matemáticos podem ser transformados em desenhos interessantes, com algumas simples instruções em BASIC. Veja os procedimentos para obtê-los nesta série de artigos.

Desenhar o gráfico de uma expressão matemática não é difícil. Basta definir uma faixa de valores para uma das variáveis da expressão e encontrar os valores correspondentes para a outra. Com um gráfico simples como $Y = X^2$, pode-se chegar a uma tabela como esta:

X	-5	-4	-3	-2	-1	0	1	2	3	4	5
Y	25	16	9	4	1	0	1	4	9	16	25

Assinalando os pontos em papel quadriculado e unindo-os com um traço, obtém-se a conhecida curva em forma de colar — o gráfico de $Y = X^2$.

Outra maneira de interpretar a curva é vê-la como o caminho percorrido por um ponto que se move ao longo de $Y = X^2$, caminho chamado de *lugar geométrico*. Manipulando e combinando diferentes lugares geométricos, podem-se produzir padrões notáveis com pouco esforço.

Para isso, vamos tomar um lugar geométrico bem conhecido — o círculo. Exprimi-lo por meio de uma fórmula é algo mais complexo do que o método anterior. Ambas as variáveis X e Y da equação são definidas em termos de uma terceira variável ou parâmetro. Variando-se tal parâmetro dentro de um conjunto dado de valores, são encontrados pares de Xs e Ys. E o círculo é representado pela seguinte equação:

X = R SENO(I)
Y = R CO-SENO(I)

Se determinarmos um conjunto de valores para X e Y enquanto o ângulo I assume valores que completem um círculo (0° a 360°), encontraremos o lugar geométrico de um círculo. O curto programa a seguir foi escrito para o CP-400.

```
10 REM DESENHAR CIRCULO
15 PMODE 4,1:SCREEN 1,0:PCLS
20 XM=256:YM=192:XC=INT(XM/2):YC=INT(XM/2)
30 R=50:PI=3.1416
40 S=PI/20
50 FOR I=0 TO 2*PI STEP S
60 PSET (XC+R*SIN(I),YC+R*COS(I),1)
70 NEXT I
```

É de notar que, alterando-se o tamanho do STEP entre os pontos demarcados (linha 50), pode-se variar a definição do círculo e a velocidade com que ele é desenhado. A maioria das versões do BASIC não são suficientemente rápidas para desenhar, a velocidade aceitável, círculos uniformes a partir de muitos pontos individuais. Para superar o problema, muitas vezes é preferível usar linhas retas para unir os pontos. De fato, com um grande número de retas chega-se ao meio-termo ideal entre a velocidade e a uniformidade do desenho.

Esse processo também pode ser utilizado para desenhar arcos e elipses. Para arcos, deve-se selecionar diferentes valores de I. Para elipses, dão-se a R valores diferentes a cada uma das fórmulas, de X e de Y. No entanto, o círculo é um fértil ponto de partida para a criação de desenhos interessantes.

No primeiro desenho traçamos uma linha a partir de um ponto fixo até todos os pontos do lugar geométrico. Os dois programas abaixo situam o ponto no centro do círculo e a sua esquerda, respectivamente.

```
10 REM FLOR
15 PMODE 4,1:SCREEN 1,0:PCLS
20 XM=256:YM=192:XC=INT(XM/2):YC=INT
   (YM/2)
30 R=50:PI=3.1416:S=PI/20
40 FOR I=0 TO 2*PI STEP S
50 X=XC+R*SIN(I):Y=YC+R*COS(I)
60 LINE(X,Y)-(XC,YC),PSET
70 NEXT I
```

```
10 REM CIRCULO E PONTO FIXO
15 PMODE 4,1:SCREEN 1,0:PCLS
20 XM=256:YM=192:XC=INT(XM/2):YC=INT
   (YM/2)
30 R=50:PI=3.1416:S=PI/20
40 FOR I=0 TO 2*PI STEP S
50 X=XC+R*SIN(I):Y=YC+R*COS(I)
60 LINE(X,Y)-(40,96),PSET:LINE(XC,YC)-
   (40,96),PSET
70 NEXT I
```

O próximo passo é usar um ponto móvel e não um fixo. Demarcam-se dois lugares geométricos simultaneamente, traçando linhas para ligar pontos correspondentes nas duas rotas. Com o programa abaixo, obtêm-se dois círculos concêntricos ligados por muitas retas.

```
10 REM CIRCULOS CONCENTRICOS
15 PMODE 4,1:SCREEN 1,0:PCLS
20 XM=256:YM=192:XC=INT(XM/2):YC=INT
   (YM/2)
30 R=50:PI=3.1416:S=PI/20
40 FOR I=0 TO 2*PI STEP S
50 X=XC+(R+30)*SIN(I):Y=YC+(R+30)*
   COS(I)
60 P=XC+R*SIN(I):Q=YC+R*COS(I)
70 LINE(X,Y)-(P,Q),PSET
80 NEXT I
```

O programa pode incorporar dados suficientes para desenhar centenas de padrões, mediante mudança de um ou de ambos os lugares geométricos. Uma variação é trocar o seno e o co-seno entre si numa das fórmulas. Outra consiste em usar potências do seno e do co-seno.

```
10 REM CIRCULOS DESALINHADOS
15 PMODE 4,1:SCREEN 1,0:PCLS
20 XM=256:YM=192:XC=INT(XM/2):YC=INT
   (YM/2)
30 R=50:PI=3.1416:S=PI/20
40 FOR I=0 TO 2*PI STEP S
50 X=XC+(R+30)*COS(I):Y=YC+(R+30)*
   SIN(I)
60 P=XC+R*SIN(I):Q=YC+R*COS(I)
70 LINE(X,Y)-(P,Q),PSET
80 NEXT I
```

Todas essas idéias comportam variações: qualquer livro de matemática fornece fórmulas para obter outras curvas. Mas tente, de preferência, criar seus próprios programas. Pequenas alterações levam o computador a fazer experiências por você.
O último programa aqui mostrado passa por um ciclo infinito de padrões gerados aleatoriamente pelo computador, embora isso não esgote as possibilidades.

```
10 REM CIRCULO E SENO
15 PMODE 4,1:SCREEN 1,0:PCLS
20 XM=256:YM=192:XC=INT(XM/2):YC=INT
   (YM/2)
30 R=50:PI=3.1416:S=PI/20
40 FOR I=0 TO 2*PI STEP S
50 X=XC+SIN(I)*80:Y=YC+(R+30)*SIN(I)
60 P=XC+R*SIN(I):Q=YC+R*COS(I)
70 LINE(X,Y)-(P,Q),PSET
80 NEXT I
```

```
10 REM CIRCULO E SENO*COS
15 PMODE 4,1:SCREEN 1,0:PCLS
20 XM=256:YM=192:XC=INT(XM/2):YC=INT
   (YM/2)
30 R=50:PI=3.1416:S=PI/20
40 FOR I=0 TO 2*PI STEP S
50 X=XC+R*SIN(I)*COS(I):Y=YC+R*SIN(I)*
   COS(I)
60 P=XC+R*SIN(I):Q=YC+R*COS(I)
70 LINE(X,Y)-(P,Q),PSET
80 NEXT I
```

```
10 REM FORMAS CIRCULARES RANDOMICAS
15 PMODE 4,1:SCREEN 1,0:PCLS
20 XM=256:YM=192:XC=INT(XM/2):YC=INT
   (YM/2)
30 R=60:PI=3.1416:S=PI/20
35 PCLS
40 C=RND(50)+1:D=RND(50)+1:E=RND(50)
   +1:F=RND(50)+1
50 FOR I=0 TO 2*PI STEP S
60 X=XC+R*SIN(I/C)*COS(I*D):Y=YC+R*SIN
   (I/E)*COS(I*F)
70 P=XC+R*SIN(I):Q=YC+R*COS(I)
80 LINE(X,Y)-(P,Q),PSET
90 NEXT I
100 A$=INKEY$:IF A$="" THEN GOTO 100
110 GOTO 35
```

Variações

Adaptar e utilizar em outras máquinas as idéias já apresentadas é muito simples. Seu micro deve ser capaz de fazer gráficos de alta resolução (pelo menos 256 x 176, de preferência), ter um BASIC de ponto flutuante com as funções SIN e COS, e comandos para desenhar pontos individuais e linhas retas, tais como o PLOT e o HPLOT do Apple. Os primeiros ajustes necessários são para XM e YM, valores máximos para X e Y que podem ser traçados em seu equipamento. Dependendo das funções que usar, você poderá decidir modificar outras constantes no programa, como R e S. Em seguida, é preciso certificar-se de que seu micro está no modo apropriado para gráficos e selecionar uma cor para o traçado. Finalmente, é necessário um comando para desenhar linhas entre as coordenadas dadas por X e Y, e P e Q. A maioria dos micros possui uma função para traçar linhas absolutas, o que torna simples essa adaptação.
Veja abaixo como os programas FLOR e CIRCULO E SEN * COS foram adaptados para o Apple.

```
10 REM *** FLOR ***
20 HGR2:HCOLOR=7
30 XM=256:YM=192:XC=INT (XM/2):YC=INT
   (YM/2)
40 R=50:PI=3.1416:S=PI/20
50 FOR I=0 TO 2*I STEP S
60 X=XC + R * SIN (I):Y = YC + R *
   COS(I)
70 HPLOT X,Y TO XC,YC
80 NEXT I
```

```
10 REM *** CIRCULO E SEN * COS ***
20 HGR2:HCOLOR = 7
30 XM = 256 : YM = 192 : XC = INT (XM/2)
   YC = INT (YM/2)
40 R = 50 : PI = 3.1416: S = PI/20
50 FOR I = 0 TO 2 * PI STEP S
60 X = XC + SIN (I) * COS (I):Y = YC +
   R * SIN (I) * COS (I)
70 P = XC + R * SIN (I):Q = YC + R *
   COS(I)
80 HPLOT X,Y TO P,Q
90 NEXT I
```

Idéias para projetos

1) No loop simples para desenhar um círculo, descrevemos como se criam arcos e elipses a partir do mesmo programa. Agora, tente encontrar uma maneira de traçar espirais.
2) Experimente usar funções como SQR (raiz quadrada) e TAN (tangente) para criar lugares geométricos. Lembre-se, no entanto, de que essas funções devem ser usadas com cuidado, pois tendem a gerar números impraticáveis.
3) Desenvolva versões animadas do programa. Recorrendo a matrizes para registrar as cinco últimas linhas traçadas, pode-se obter um conjunto de cinco linhas que "se perseguem".
4) Que tal criar padrões baseados em três lugares geométricos? Faça dois deles simples (círculo e linha, por exemplo), para evitar que a imagem resulte muito confusa.

MICROS SHARP

Equipados com microprocessador de 8 bits, ROM, RAM, teclado alfabético e conexões para diversos periféricos, o PC-1251 e o PC-1500A, da Sharp, são verdadeiros microcomputadores de bolso.

Sharp PC-1251

O PC-1251, o menor computador existente no mercado, mede 135 x 70 x 10 mm e pesa apenas 115 g. O teclado alfabético tem a metade do tamanho de um teclado comum, o que exige manuseio cuidadoso, pois as teclas têm apenas 4 mm de largura. O PC-1251 apresenta também um teclado numérico comum, com teclas maiores, que permitem usar o micro como calculadora de bolso. O visor de cristal líquido (LCD) mostra uma linha de 24 caracteres por vez, que pode ser movimentada para os lados, e também na vertical, no modo de programação. Mediante um pequeno botão lateral, faz-se o ajuste angular da tela, de modo a adequá-la às condições de iluminação reinantes.

O PC-1251 funciona em três modos, acionados por uma chave seletora: modo de programação (PRO); de atribuição de funções a determinadas teclas (RSV); e modo RUN, para executar os programas ou utilizar a calculadora. Para armazenar e imprimir programas e dados, o PC-1251 conta com uma unidade à parte, a impressora/microcassete CE-125, na qual se encaixa o microcomputador.

A versão do BASIC, bastante eficiente considerando-se o tamanho da máquina, contém funções úteis, como ASC e CHR$, embora não possua a opção ELSE para o comando IF-THEN. Outra desvantagem é a utilização dos mesmos nomes, de uma letra, para as variáveis numéricas e os strings: se A for usado para um número, A$ não servirá para o string. As mensagens de erro reduzem-se a nove e expressam-se apenas como uma letra, o que dificulta a depuração dos programas. Em cada programa, a numeração das linhas vai até 999. Para utilizá-lo, é necessário digitá-lo novamente, a menos que se disponha de uma unidade de microcassete adicional. A memória de 4 Kbytes comporta um programa por vez, e mesmo dividindo-os em subprogramas separados por END e executados por um GOTO, esse recurso só serve para programas extremamente reduzidos.

Sharp PC-1500A

Três vezes mais pesado (375 g) e com dimensões maiores, o PC-1500A é superior ao PC-1251 em termos de teclado, BASIC e possibilidades de conexão a periféricos. Seu visor LCD mostra 26 caracteres, sem ajuste angular para melhor visibilidade. Este micro dispõe de seis teclas adicionais programáveis para até dezoito funções. As teclas alfabéticas da fileira superior já vêm programadas com comandos BASIC: as palavras-chave não estão escritas nas teclas, mas indicadas numa máscara plástica.

O BASIC Microsoft deste modelo é superior ao do PC-1251. As variáveis podem ter duas letras e o mesmo nome serve para variáveis numéricas e strings; conta ainda com recursos gráficos e sonoros, além de comandos ON ERROR GOTO, TRON e TROFF e um relógio-calendário residente, de funcionamento permanente. Há 39 mensagens de erro para a máquina padrão e outras dezesseis usadas com os periféricos adicionais — estas expressam-se apenas sob a forma de números, o que dificulta a depuração.

A RAM de 8,5 Kbytes pode ser expandida de várias maneiras. Na parte inferior da máquina encaixa-se um cartucho de expansão de memória, oferecido em dois tipos: volátil (só mantém os dados enquanto acoplado à máquina), com 4 ou 8 Kbytes; e não volátil (acionado a pilha), com 8 ou 16 Kbytes.

Outro recurso adicional do PC-1500A, disponível no mercado, é uma interface dotada de saída paralela e serial para comunicação com impressoras e computadores até de grande porte. Encontra-se também no mercado o periférico "tablete de software" (Software Board), que contém 140 teclas tipo membrana, programáveis pelo usuário para executar diferentes tarefas, como totalizar colunas de números e chamar sub-rotinas.

Com seu BASIC bastante poderoso e recursos gráficos e de comunicação, este micro de bolso da Sharp ressente-se apenas da ausência de software comercialmente disponível. O fabricante oferece uma fita com uma seleção de programas. Para utilização com impressora, o PC-1500A conta com uma série de comandos BASIC, que controlam a produção e a impressão de imagens.

O micro é acompanhado de um livro com listagens de 53 programas, que permitem aplicações em diversas áreas: matemática, estatística, eletricidade, comércio e jogos.

Compartimento para pilhas
Contém as quatro pilhas necessárias para acionar o computador.

Microprocessador
Projetado para o PC-1500, com endereçamento em 8 bits.

Saída para cartucho
Interface que permite o acoplamento de cartuchos de RAM adicional. Acopla-se aqui também a unidade impressora-cassete, ou uma interface serial.

Entrada para alimentação externa
O computador também é acionado por corrente alternada, usando-se um transformador.

ROM com BASIC
Este chip contém o BASIC Microsoft utilizado no computador.

Periféricos Sharp
O Sharp modelo PC-1251 encaixa-se na unidade adicional CE-125, dotada de gravador de microfita e impressora térmica, que imprime 24 caracteres por linha. A unidade mede 205 x 149 x 23 mm.

O PC-1500A dispõe de grande variedade de periféricos. A unidade CE-150, por exemplo, contém uma impressora-plotter com caracteres de nove tamanhos, recursos para imprimir gráficos a quatro cores e cartesianos. Conta com interface para dois gravadores cassete. A interface CE-158 padrão RS232C e paralela permite conexão com quase todos os tipos de micro e mesmo computadores de grande porte.

Os módulos de memória adicional CE-151, CE-155, CE-159 e CE-161 são cartuchos contendo RAM do tipo CMOS, de 2 a 16 Kbytes, alguns não voláteis.

Chips de RAM
Fornecem 8,5 K de memória, dos quais 6,6 são disponíveis para o usuário.

Chips de tela
Estes quatro chips organizam os dados para exibição no visor de cristal líquido.

Placa principal
Para reduzir o tamanho do micro, a placa foi dividida em duas partes, conectadas por dois cabos flexíveis. São poucos os componentes CMOS, que realizam múltiplas funções.

Entrada/saída
Este chip controla os periféricos externos, como a unidade impressora-cassete.

SHARP PC-1251

MEMÓRIA
4 K de RAM.

TELA
Visor LCD, 1 x 24 caracteres.

TECLADO
Alfabético e numérico.

LINGUAGENS
BASIC simples.

PERIFÉRICOS
Unidade com microfita e impressora térmica com 1 x 24 caracteres.

SHARP PC-1500A

MEMÓRIA
8,5 K de RAM, expansível por cartuchos.

TELA
Visor LCD, 1 x 26 caracteres.

TECLADO
Alfabético e numérico, com seis teclas programáveis para dezoito funções.

LINGUAGENS
BASIC com muitos recursos.

PERIFÉRICOS
Unidade com impressora-plotter a quatro cores e saída para dois gravadores cassete. Interface para comunicação com outros micros.

Sharp PC-1251

Pesando apenas 115 g, esta calculadora com recursos de micro é útil para engenheiros e cientistas. O pequeno tamanho das teclas, no entanto, não é adequado para as tarefas que exigem muita digitação de texto.

Sharp PC-1500A

Este micro de bolso conta com um BASIC poderoso e muitos recursos úteis, como relógio-calendário permanente, impressora a quatro cores e saída para comunicação com outros micros e computadores de grande porte.

FERRAMENTAS ESSENCIAIS

Pequenos consertos podem ser feitos em casa. O domínio das técnicas de soldagem, por exemplo, muitas vezes evita que o principiante recorra aos dispendiosos serviços especializados em micros.

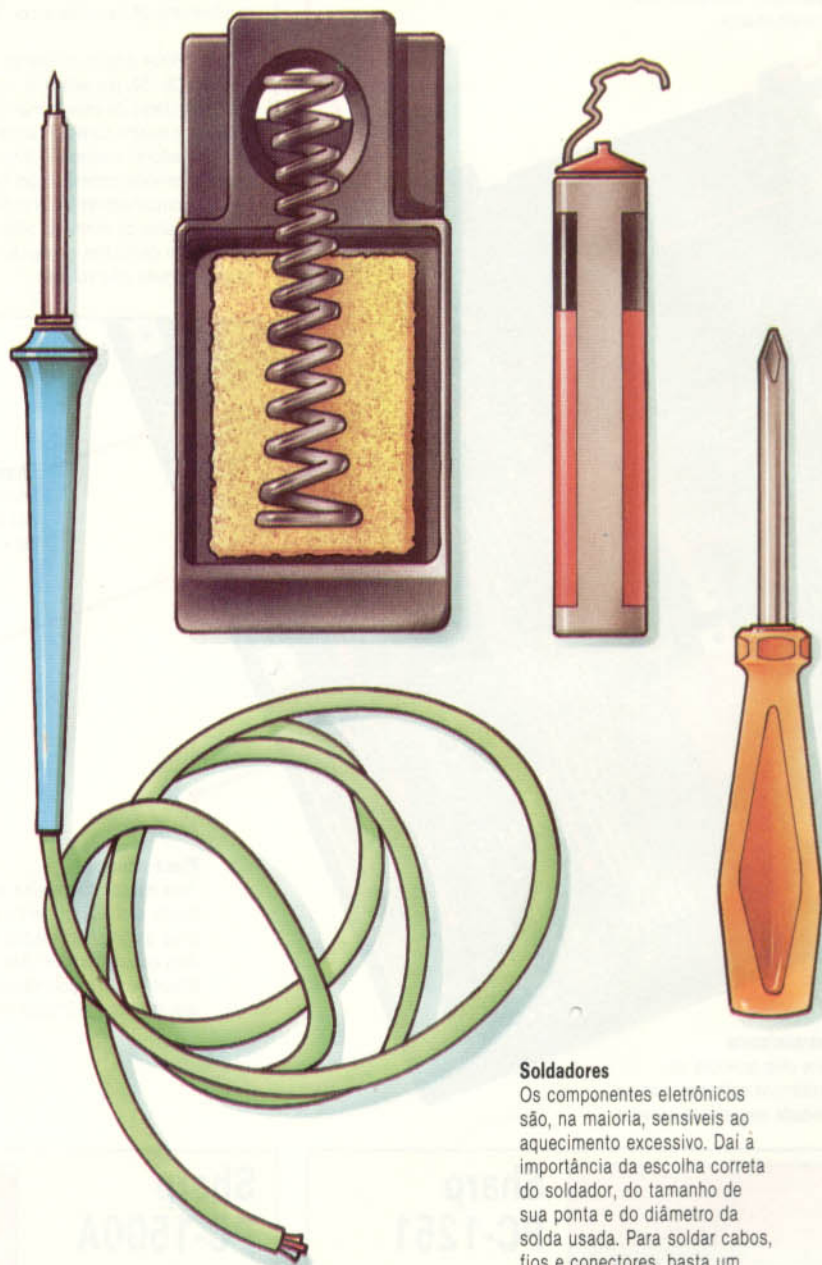
O processo de soldagem consiste em juntar dois objetos de metal por meio de uma liga metálica branda — que se funde com facilidade. Para trabalho em eletrônica, emprega-se uma liga de chumbo e estanho.

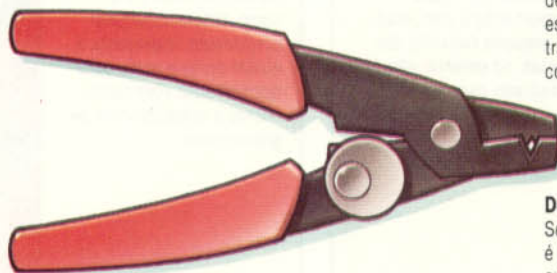
De início, os objetos devem ser aquecidos, por meio do aparelho de solda, a temperatura superior ao ponto de fusão da liga (280°C).

Em seguida, aplica-se a solda ao componente (e não à ponta do soldador) e retira-se o aparelho. Ao esfriar, a solda solidifica e a junção está feita.

Quando se aplica o soldador sobre o fio de solda, a liga derrete com muita rapidez. Contudo, ela retorna ao estado sólido assim que entra em contato com o componente frio. O resultado é insatisfatório. No melhor dos casos, não haverá junção; no pior, a ligação ficará muito fraca, impedindo o fluxo contínuo de eletricidade. A única forma de evitar que isso aconteça é aquecer o componente até que a solda derreta.

Ferramentas de bancada

Para trabalhar com os minúsculos componentes dos computadores, encontra-se à venda uma série de tornos, pinças e prendedores, a custo relativamente baixo. Se, com tal equipamento, o aperto for ainda insuficiente, tente fazer calços usando fita adesiva, com o lado aderente para fora, de modo a fixar melhor o componente na pinça.

Soldadores

Os componentes eletrônicos são, na maioria, sensíveis ao aquecimento excessivo. Daí a importância da escolha correta do soldador, do tamanho de sua ponta e do diâmetro da solda usada. Para soldar cabos, fios e conectores, basta um soldador de 15 a 25 W e solda de uso geral de 1,5 mm. Por ser demorado seu aquecimento, esses aparelhos permitem trabalhar com segurança em componentes delicados.

Descascador de fios

Sempre que usar cabos ou fios, é preciso descascar com cuidado toda a cobertura e o isolante. Descascadores de fios simples, como o da figura, custam pouco e podem ser pré-ajustados a várias bitolas de fio, de modo a remover o isolante, deixando o fio intato.

Soldagem de plugue

Descascar e estanhar
O primeiro estágio do processo de soldagem de um fio a um plugue consiste em descascar os fios do cabo, removendo o isolante. Retire mais fio que o necessário e, depois, corte no tamanho certo. Fixe o cabo no torno ou na pinça e aqueça-o com o soldador. Aplique a solda ao cabo e, quando começar a derreter, use o soldador para recobrir com a solda toda a parte exposta do cabo. Esse processo, chamado estanhamento, torna mais simples a soldagem — nesse ponto, a solda está no componente.
Repita o mesmo procedimento no plugue. Enquanto o terminal está quente o bastante para derreter a solda, encoste o cabo estanhado e só retire o soldador quando ambas as soldas (do componente e do cabo) estiverem fundindo. Depois, sopre-as para resfriar mais depressa e está pronta a junção.

Acabamento
O último estágio do trabalho consiste em remover as sobras de material. Corte os cabos bem curtos, assim como os pinos terminais nos componentes — mas antes teste o funcionamento. A razão para cortar é simples: pontas longas sobrando no cabo ou pinos do terminal protuberantes podem entrar em contato com outra peça e causar curto-circuito ou, pior, um desses irritantes defeitos de ocorrência intermitente.

Chaves adequadas
Há dois tipos de chave de parafuso: de fenda e em cruzeta. Esta última pode ter ponta Philips ou Pozidriv. Na maioria dos micros, serve qualquer um dos dois tipos. Convém, no entanto, dispor de ambos em vários tamanhos.

Alicates
O principiante precisa de apenas dois tipos de alicate — o de bico chato, para serviços pesados, como cortar fios grossos, e o de ponta, para tarefas que exigem toque mais leve.

Alicate de corte
Entre os alicates de corte, os principais são os de corte lateral, como o da ilustração, e os de corte de ponta, cujas lâminas formam um ângulo de 90° com o plano de rotação da ferramenta. Numa emergência, porém, um cortador de pregos pode resolver o problema.

DUPLO DESEMPENHO

Introduzindo-se dados variáveis e a técnica matemática da recorrência (uma instrução que chama a si mesma) nas rotinas que executam seqüências de comandos, o LOGO consegue efeitos interessantes.

RECURSION

Por este termo, traduzido como "recorrência", indica-se uma série de rotinas repetidas dentro de um programa, na qual o resultado de cada repetição depende do resultado da repetição anterior. A rotina chama a si própria até a tarefa estar completa.

Na linguagem LOGO, uma palavra — como TAMANHO, por exemplo — pode ser empregada de três maneiras distintas. A diferença entre tais usos é assinalada pelas seguintes notações: TAMANHO, :TAMANHO ("dois pontos tamanho") e "TAMANHO ("aspas tamanho").

Ao encontrar a palavra TAMANHO sem nenhuma pontuação, o LOGO a considera o nome de uma rotina e executa os comandos da definição de TAMANHO. :TAMANHO é usada para designar o valor atribuído ao nome da variável — se o LOGO encontrar :TAMANHO, ele irá procurar o valor associado a esse nome. "TAMANHO refere-se aos procedimentos e nomes de variáveis, mas indica apenas seus nomes, e não seus valores.

Portanto, "TAMANHO refere-se a uma variável, enquanto :TAMANHO designa o valor atribuído a essa variável. (Observe que as aspas são colocadas antes, e não depois das palavras. O final do nome é indicado por um espaço.)

Na versão MLOGO, depois dos comandos EDITE, SEM e LISTAR, o nome de um procedimento é escrito sem aspas. Desse modo, a sintaxe correta é EDITE QUADRADO, mesmo que QUADRADO não seja uma chamada para o procedimento QUADRADO, mas apenas o nome desse procedimento.

Quando você quiser empregar qualquer dos procedimentos definidos até agora, digite o nome do procedimento, como faria com qualquer comando, como DESENHE ou SEMT. No entanto, o uso de outros comandos — FRENTE, por exemplo — exige informações adicionais. Por si mesma, a palavra FRENTE nada significa — é preciso atribuir-lhe um valor para que o LOGO execute o comando. Incluindo variáveis em nossos procedimentos, podemos introduzir qualquer valor desejado e variar o efeito obtido quando se chama o procedimento.

O procedimento para desenhar um quadrado exemplifica essa variação:

```
APRENDA QUADRADO
  REPITA 4 [FRENTE 50 DIREITA 90]
FIM
```

Como está, o procedimento traça um quadrado cujos lados medem 50 unidades. Contudo, ele seria muito mais útil se o tamanho do quadrado pudesse ser modificado de acordo com valores escolhidos. Para fazer com que QUADRADO aceite esse tipo de dado, use os recursos de edição, a fim de substituir o valor fixo 50 pela variável "LADO e acrescente :LADO à linha do título. Com as modificações, o procedimento ficará assim:

```
AP QUADRADO :LADO
  REPITA 4 [FR :LADO DI 90]
FIM
```

Agora, quando chamar um procedimento, você precisa atribuir um valor à variável "LADO. Experimente QUADRADO 40, QUADRADO 10 etc. para ver como varia o tamanho da figura.

Veja o que acontece quando se digita QUADRADO 30. A primeira coisa que o LOGO faz é consultar a definição de QUADRADO. A linha do título informa que é necessária a introdução de um dado, e que ele se chamará "LADO. O valor na linha de entrada — nesse caso, 30 — é, portanto, atribuído à variável "LADO. A seguir, são obedecidos os comandos incluídos na definição do procedimento.

A melhor maneira de se entender o processo é imaginar que os nomes de variáveis remetem a caixas contendo valores. Quando chega à linha FRENTE :LADO, o LOGO procura a caixa intitulada "LADO, lê o valor ali encontrado e o utiliza como um dado para o comando FRENTE. A caixa "LADO é usada apenas nesse procedimento. Se algum outro também utilizar "LADO como nome de uma variável, usará outra caixa. Portanto, "LADO é uma variável *local*.

Os dados variáveis também podem ser utilizados em subprocedimentos. O procedimento CASA é uma das soluções do exercício proposto no artigo anterior. Ele pode ser modificado de modo a aceitar valores diferentes:

```
AP CASA :GRANDE
  QUADRADO :GRANDE
  FR :GRANDE DI 30
  TRIANGULO :GRANDE
  ESQUERDA 30 VOLTE :GRANDE
FIM
AP QUADRADO :TAMANHO
  REPITA 4 [FR :TAMANHO DI 90]
FIM
AP TRI :LADO
  REPITA 3 [FR :LADO DI 120]
FIM
```

Três nomes diferentes de variáveis constam desses procedimentos — "GRANDE, "TAMANHO e "LADO. Pode-se usar o mesmo nome para as três variáveis, já que elas se restringem aos procedi-

mentos nos quais são empregadas. Mas isso dá margem a muita confusão.

Para entender como funciona esse procedimento, examine o que acontece quando se digita CASA 30. O LOGO lê a linha de entrada e atribui o valor 30 à variável "GRANDE, em CASA. Assim, a primeira linha de CASA equivale agora a QUADRADO 30. À variável "TAMANHO, em QUADRADO, atribui-se o valor 30. Quando QUADRADO é executado, FRENTE :TAMANHO torna-se FRENTE 30. Procedimento semelhante é seguido no caso de TRI.

Experimente adaptar os procedimentos para o traçado de um tabuleiro de 5 x 5 quadrados, de modo que o tamanho do quadrado seja um dado de entrada.

A seguir, apresentamos um procedimento para o desenho de polígonos no qual o número de lados varia:

```
AP POLI :LADOS
  REPITA :LADOS [FR 50 DI 360/ :LADOS]
FIM
```

Nesse procedimento com uma variável, POLI 3 traçará um triângulo, POLI 4 um quadrado, e assim por diante. Contudo, em todos os polígonos, os lados terão 50 unidades de comprimento.

Um procedimento mais geral, que trace polígonos de qualquer tamanho, requer duas variáveis — uma para o número de lados e outra para o comprimento dos lados. Adapte o procedimento POLI de modo que 50 seja substituído por um nome de variável, acrescentado à linha do título.

```
AP POLI :LADOS :TAMANHO
  REPITA :LADOS [FR :TAMANHO DI 360/ :LADOS]
FIM
```

Assim, POLI 5 30 traçará um pentágono, cujos lados medem 30 unidades. De modo semelhante, você pode modificar sua nova versão do procedimento do tabuleiro, para que ele desenhe tabuleiros de qualquer tamanho (e não apenas de 5 x 5 quadrados). Empregue duas variáveis — o número de quadrados em cada direção (horizontal e vertical) e o tamanho deles.

Variáveis globais

Além das variáveis locais em relação aos procedimentos que as utilizam, é possível definir variáveis que sejam usadas por todos os procedimentos. Conhecidas como "globais", são úteis para se estabelecer comunicação entre procedimentos diferentes. Mas seu emprego apresenta a desvantagem de tornar mais difícil a depuração de erros e, por isso, elas devem ser usadas com critério.

Usa-se o comando FACA para atribuir valores às variáveis globais. FACA "LADO 3 significa que o valor 3 está sendo associado à variável "LADO. FACA "LADO :LADO + 1 aumenta em 1 o valor de "LADO. Nesse exemplo, o significado preciso da notação é: encontre o valor da variável "LADO, adicione 1 e atribua o resultado à própria variável "LADO. Em ambos os casos, FACA exige a entrada de dois dados — o nome da variável e o valor que será associado a ela.

Como resumo dos recursos de programação tratados neste artigo, criamos alguns procedimentos para o traçado de espirais. O procedimento principal receberá o nome de ESPIRAL. Ele opera com três variáveis: o comprimento inicial da linha a ser traçada, o ângulo em que a tartaruga deve girar a cada rotação da espiral e um fator de escala pelo qual o comprimento inicial deve ser multiplicado (para se produzir o efeito de espiral). Conjuntos diferentes de valores podem ser usados para se obter efeitos diferentes — nós experimentamos com 70 283 0.95, 70 143 0.95 e 20 243 1.05. Tente outros conjuntos de números e veja o que acontece.

FECHADO é um novo comando. Ele interrompe a tartaruga e emite uma mensagem de erro, quando ela transpõe as margens da tela. Em muitos casos, a ultrapassagem dos limites da tela produz resultados interessantes. No entanto, a rotina que estamos estudando exige o uso de FECHADO, pois a ultrapassagem das margens da tela estraga o efeito de espiral.

O procedimento principal ESPIRAL desenha repetidas vezes uma linha, cujo comprimento é determinado pelo fator escalar. A seguir, ele gira

Traçado da espiral

```
AP E.FRENTE :NUMERO
  FRENTE :ESCALA
FIM
AP ESPIRAL :LADO :ANGULO :FATOR
  INICIO
  REPITA 1000 [E.FRENTE :LADO DI :ANGULO
    AUMENTO :FATOR]
FIM
AP INICIO
  DESENHE FECHADO FACA "ESCALA :LADO
FIM
AP AUMENTO :DIST
  FACA "ESCALA :ESCALA * :DIST
FIM
```

Exercício 3

A) Escreva um procedimento para traçar um círculo de raio 50. Modifique o procedimento de modo que o raio possa ser variável.
B) Escreva o procedimento para o traçado de um "alvo" (cinco círculos concêntricos).

Registre

Linha Sinclair

Como usar muitas vezes a mesma função dentro de um programa, sem redigitá-la? Defina a função como um string relacionado a uma variável alfanumérica:

```
LET A$ = "X**2+Y**3-1"
```

Depois, para utilizar a função, basta pedir o valor da variável:

```
PRINT VAL A$
```

Esse recurso pode ser adaptado sem dificuldade para qualquer outro tipo de microcomputador.

Respostas do exercício 2

A) Quebra-cabeça Tangram

O homem sentado, o homem rezando e o gato usam o lado do paralelogramo oposto ao lado usado pelo cão no problema anterior. Em LOGO, para inverter uma forma, basta transformar os comandos DIREITA em ESQUERDA, e vice-versa. Assim, em vez de formar o paralelogramo como:

```
AP PAR
   REPITA 2 [FR 25 DI 45 FR 35 DI 135]
```

nós usaríamos um paralelogramo assim:

```
AP PAR1
   REPITA 2 [FR 25 ES 45 FR 35 ES 135]
```

Homem correndo

```
AP CORRENDO
 MOVE1 TRI1 MOVE2 PAR MOVE3 TRI3 MOVE4
 TRI3 MOVE5 QUADRADO MOVE6 TRI1 MOVE7
 TRI2 MOVE8
FIM
AP MOVE1
 ES 45
FIM
AP MOVE2
 SC FR 25 DI 135 FR 17.5 ES 45 CL
FIM
AP MOVE3
 SC FR 75 DI 90 CL
FIM
AP MOVE4
 SC DI 90 FR 25 DI 90 CL
FIM
AP MOVE5
 SC FR 50 DI 135 FR 50 DI 135 CL
FIM
AP MOVE6
 SC DI 135 FR 21 DI 135 FR 25 ES 90
 FR 50 ES 90 FR 25 DI 90 CL
FIM
AP MOVE7
 SC FR 25 DI 135 FR 71 DI 45 VO 35 CL
 FIM
AP MOVE8
 SC FR 35 ES 90 FR 25 DI 45 FR 17.5
 ES 45 FR 25 DI 135 CL
FIM
```

Homem sentado

```
AP SENTADO
 MOVE1 TRI1 MOVE2 TRI2 MOVE3 TRI3 MOVE4
 TRI1 PAR1 MOVE5 QUADRADO MOVE6 TRI3
 MOVE7
FIM
AP MOVE1
 ES 45
FIM
AP MOVE2
 SC FR 25 ES 45 FR 17.5 DI 90 CL
FIM
AP MOVE3
 SC VO 15 ES 90 CL
FIM
AP MOVE4
 SC FR 50 DI 45 FR 25 DI 90 CL
FIM
AP MOVE5
 SC FR 25 ES 45 FR 35 ES 45 CL
FIM
AP MOVE6
 SC VO 50 ES 90 CL
FIM
AP MOVE7
 SC VO DI 135 VO 50 DI 90 FR 35
 ES 90 CL
AP MOVE8
 SC FR 35 ES 90 FR 25 DI 45 FR 17.5
 ES 45 FR 25 DI 135 CL
FIM
```

Homem rezando

```
AP REZANDO
 MOVE1 TRI1 MOVE2 PAR1 MOVE3 TRI3 MOVE4
 TRI3 MOVE5 TRI1 MOVE6 TRI2 MOVE7
 QUADRADO MOVE8
FIM
AP MOVE1
 ES 90
FIM
AP MOVE2
 SC FR 25 DI 135 FR 30 CL
FIM
AP MOVE3
 SC ES 45 FR 35 ES 135 VO 50 CL
FIM
AP MOVE4
 SC DI 90 FR 50 ES 135 CL
FIM
AP MOVE5
 SC DI 90 FR 50 ES 135 FR 5 DI 90 CL
FIM
AP MOVE6
 SC DI 90 FR 25 DI 45 FR 7.5 DI 45
 VO 35 CL
FIM
AP MOVE7
 SC FR 35 DI 135 FR 7.5 ES 45 FR 55
 DI 45 CL
FIM
AP MOVE8
 SC ES 45 FR 36 DI 45 FR 56 ES 135
 FR 5 ES 45 VO 25 CL
```

Gato

```
AP GATO
 MOVE1 TRI3 MOVE2 QUADRADO MOVE3 TRI1
 MOVE4 TRI1 MOVE5 TRI3 MOVE6 PAR1
 MOVE7 TRI2 MOVE8
FIM
AP MOVE1
 SC FR 50 DI 90 CL
FIM
AP MOVE2
 DI 170
FIM
AP MOVE3
 SC DI 90 FR 25 ES 90 CL
FIM
AP MOVE4
 DI 180
FIM
AP MOVE5
 SC DI 90 FR 25 ES 80 FR 50 DI 45
 FR 50 DI 90 CL
FIM
AP MOVE6
 ES 155
FIM
AP MOVE7
 SC ES 160 FR 35 CL
FIM
AP MOVE8
 SC FR 35 ES 45 FR 21 DI 135 CL
FIM
```

B) Casa

```
AP CASA
 QUADRADO FR 50 DI 30
 TRI ES 30 VO 50
FIM

AP QUADRADO
 REPITA 4 [FR 50 DI
 90]
FIM

AP TRI
 REPITA 3 [FR 50 DI
 120]
FIM
```

C) Quadro

```
AP QUADRO
 REPITA 5 [LINHA FR
 10] VO 50
FIM
AP LINHA
 REPITA 5[ QUADRADO DI
 90 FR 10 ES 90] ES 90
 FR 50 DI 90
FIM
AP QUADRADO
 REPITA 4 [FR 10 DI 90]
FIM
```

D) Estrela de seis pontas

```
AP SEIS.ESTRELAS
 TRI MOVE TRI
 MOVE.VOLTA
FIM
AP TRI
 DI 30 REPITA 3
 [FR 50 DI 120]
FIM
AP MOVE
 SC FR 29 DI 60 CL
FIM
AP MOVE.VOLTA
 SC ES 60 VO 29 CL
FIM
```

a tartaruga num ângulo fixo; e, por fim, altera o fator escalar.

O comprimento das linhas traçadas aumenta ou diminui conforme o fator escalar seja maior ou menor do que 1. O número elevado depois de REPITA serve apenas para manter o procedimento rodando por bastante tempo. Se você acha que já viu suficientemente o procedimento, acione [CONTROL]-G para interrompê-lo.

Todas as variáveis são locais, com a exceção de "ESCALA. Ela é uma variável global porque "AUMENTO muda seu valor, e este novo valor deve estar disponível para E.FRENTE. "ESCALA serve, portanto, para a comunicação entre dois procedimentos.

Recurso à recorrência

Um dos primeiros programas definidos nesta seção foi o procedimento para o desenho de um quadrado. Comandada por ele, a tartaruga movia-se para a frente um certo número de unidades, girava 90° para a direita, e repetia três vezes essas operações. Apresentamos em seguida outra maneira de se desenhar um quadrado:

```
AP QUADRADO
  FR 50
  DI 90
  QUADRADO
FIM
```

Se você testar esse procedimento, verá que a tartaruga traça um quadrado e depois continua a se movimentar pelo perímetro da figura, até que se tecle [CONTROL]-G. A característica mais evidente desse novo procedimento QUADRADO é que ele "chama" a si mesmo — em outros termos, ele é recorrente (ou recursivo).

Quando se roda esse procedimento, o LOGO consulta a definição de QUADRADO e passa a obedecer às instruções. A tartaruga move-se para a FRENTE 50 unidades e depois gira 90° para a DIREITA.

A instrução seguinte é QUADRADO. O LOGO volta portanto a consultar a definição e começa a segui-la. Isso continuará se repetindo indefinidamente se o programa não for interrompido.

As chamadas recorrentes também são usadas em procedimentos que requerem dados:

```
AP POLI :LADO :ANGULO
  FR :LADO
  DI :ANGULO
  POLI :LADO :ANGULO
FIM
```

Esse procedimento traça inúmeros polígonos. Como exercício, você poderia utilizar no procedimento um ângulo de 89°. Outra possibilidade é a mudança do valor do dado na chamada recursiva. Assim:

```
AP POLIESPI :LADO :ANGULO
    FR :LADO
    DI :ANGULO
    POLIESPI (:LADO + 5) :ANGULO
FIM
```

A única diferença entre este procedimento e o anterior é que há um acréscimo de 5 unidades ao valor de LADO cada vez que ele é chamado. Portanto, se você começar com POLIESPI 10 90, a primeira chamada traçará uma linha de 10 unidades de comprimento; a segunda traçará um linha de 15; a terceira, de 20; e assim por diante. O resultado será uma espiral. Experimente com dados diferentes: comece com 10 90, 10 95, 10 120, 10 117, 10 144 e 10 142. Você também pode modificar o procedimento transformando a adição em subtração ou multiplicação.

Mostraremos agora um procedimento que aumenta o ângulo, em vez de aumentar o valor do lado:

```
AP ESPIRAL :LADO :ANGULO :AUM
  FR :LADO
  DI :ANGULO
  ESPIRAL :LADO (:ANGULO + :AUM) :AUM
FIM
```

Experimente com diversos valores: 5 0 7, 10 40 30, 15 2 20, 5 30 20. Por que algumas figuras são fechadas e outras não? Você poderia encontrar uma regra para isso?

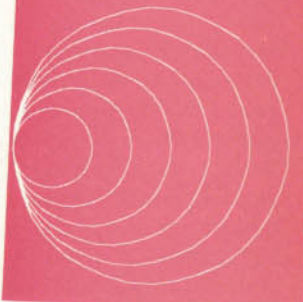
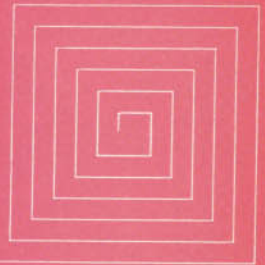
A repetição simples de um trecho de código é conhecida como "iteração". O LOGO usa REPITA com essa finalidade, enquanto outras linguagens empregam outros comandos, tais como FOR-NEXT, REPEAT-UNTIL e WHILE-WEND. Se você já programou em outras linguagens, talvez sinta dificuldade para romper o hábito de utilizar a iteração. Mas valerá a pena, pois os grafismos do LOGO são ideais para a experimentação com as chamadas recorrentes.

Regras de interrupção

Os procedimentos recursivos citados continuam a se repetir indefinidamente. Assim, precisamos dispor de um meio para interrompê-los em dado instante. Tomando como exemplo o procedimento QUADRADO, poderíamos interrompê-lo logo após o traçado do primeiro quadrado completo, quando a direção da tartaruga estiver de volta em 0. Isso pode ser feito acrescentando-se uma "regra de interrupção" ao procedimento:

```
AP QUADRADO :LADO
  FR :LADO
  DI 90
  SE DIRECAO = 0 ENTAO PARE
  QUADRADO :LADO
FIM
```

Os novos comandos primitivos são PARA e SE. O primeiro interrompe a execução de um procedimento e devolve o controle para o procedimento que o chamou. A declaração SE é o modo pelo qual o LOGO toma decisões. SE é seguido por uma condição, e ENTAO, por uma ação realizada *se* a condição for verdadeira.

Vamos examinar uma versão de POLIESPI e ver em detalhes o que ocorre quando nela se introduz uma regra de interrupção:

```
AP POLIESPI :COMPRIMENTO
  SE :COMPRIMENTO > 15 ENTAO PARE
  FR :COMPRIMENTO
  DI :90
  POLIESPI (:COMPRIMENTO – 5)
FIM
```

Isso é o que acontece quando rodamos POLIESPI 10. O procedimento POLIESPI é chamado e associa-se o valor 10 a uma variável local. Como tal valor não supera 15, o LOGO passa a executar o movimento FRENTE 10 DIREITA 90. Depois, chama POLIESPI de novo, mas desta vez o valor da variável é 15, o que leva a nova chamada de uma cópia do procedimento. Como COMPRIMENTO não ultrapassa 15, a tartaruga é movida para a FRENTE 15 DIREITA 90 e, então, realiza-se nova chamada para POLIESPI.

Contudo, a variável local foi aumentada para 20, de modo que o procedimento pára e devolve o controle ao procedimento que o chamou (POLIESPI 15). Este, por sua vez, ao chegar ao final, devolve o controle a seu procedimento de chamada. Quando este termina, o programa se encerra.

A recorrência no LOGO opera com procedimentos que chamam cópias de si mesmos. As chamadas recursivas são cópias que existem lado a lado com o procedimento original, embora funcionem como se fossem diferentes deste.

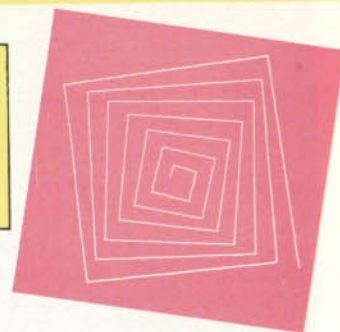
Quando chegam ao final, essas cópias sempre transferem o controle ao procedimento que as chamou. Para ilustrar esse processo de retorno a partir das chamadas de procedimento, podemos reescrever POLIESPI do seguinte modo:

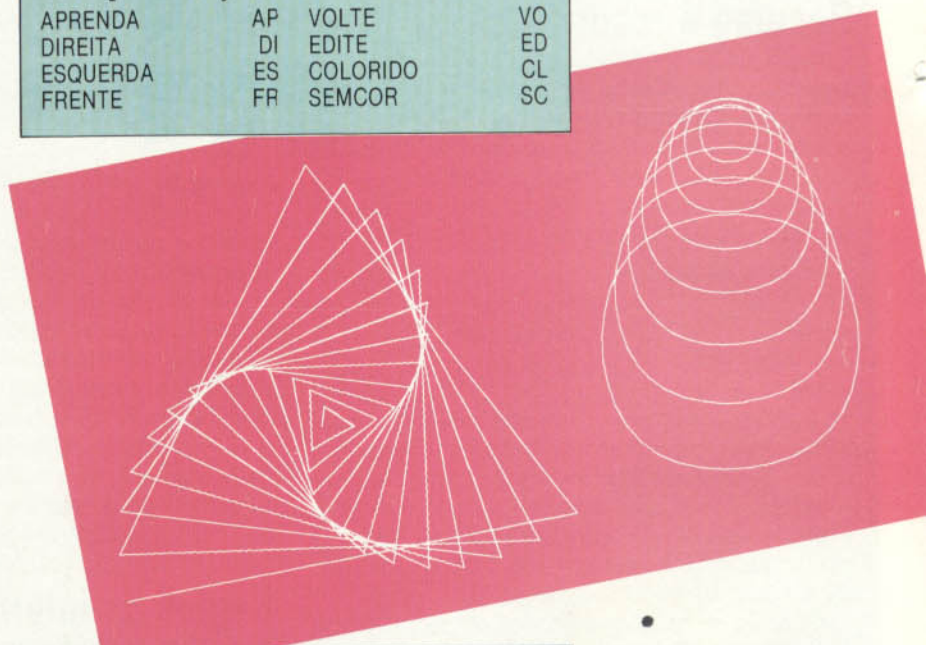
```
AP POLIESPI :COMPRIMENTO
  SE :COMPRIMENTO > 50 ENTAO PARE
  POLIESPI (:COMPRIMENTO + 15)
  FR :COMPRIMENTO
  DI 90
FIM
```

Ao rodar esse procedimento, você perceberá que a figura é desenhada "de trás para a frente": as linhas da espiral são traçadas da margem para o centro, e não o contrário. (Isso ficará mais claro se for usado um valor mais alto na sentença condicional — por exemplo, 250 em vez de 50.) O LOGO traça cada linha à medida que o controle é transferido pelas chamadas de procedimento. Nos exemplos anteriores, uma linha era traçada e então o controle passava a outro procedimento. Aqui, todos os procedimentos são chamados antes do início de qualquer desenho. Assim, usa-se antes o último valor de COMPRIMENTO introduzido.

Recursion (o termo mais usual para indicar a recorrência) é uma técnica que ocupa muito espaço na memória. Os procedimentos implementados com mais eficiência são aqueles em que a chamada recursiva está na última linha. Independente de quantas vezes sejam chamadas, elas não necessitam de memória adicional. Portanto, um procedimento recorrente só vale o trabalho de ser implementado quando é deste tipo.

Exercício 4

Escreva um procedimento para o desenho de uma torre formada de quadrados superpostos, de modo que a cada chamada o tamanho dos lados seja reduzido à metade.

Simplificação

APRENDA	AP	VOLTE	VO
DIREITA	DI	EDITE	ED
ESQUERDA	ES	COLORIDO	CL
FRENTE	FR	SEMCOR	SC

Respostas do exercício 3

A) Círculo

Este procedimento desenha um círculo, tendo o raio como um dado variável, e onde a posição inicial da tartaruga é um ponto na circunferência:

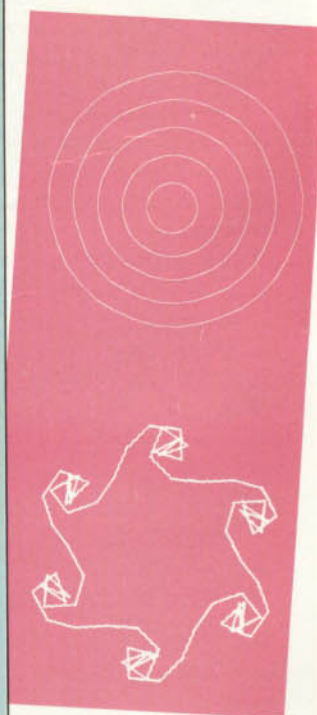
```
AP CIRCULO :RAIO
  REPITA 36 [FR(2* :PI* :RAIO/36) DI 10]
FIM
FACA "PI (3.14159)
```

Se modificamos esse procedimento de modo que a posição inicial fique no centro do círculo, obtemos:

```
AP CENTRO.CIRCULO :RAIO
  SC ES 90 FR :RAIO DI 90 CL
  CIRCULO :RAIO
  SC ES 90 VO :RAIO DI 90 CL
FIM
```

B) Alvo

O ALVO emprega CENTRO.CIRCULO para desenhar cinco círculos concêntricos:

```
AP ALVO
  CENTRO.CIRCULO 10 CENTRO.CIRCULO 20
CENTRO.CIRCULO 30 CENTRO.CIRCULO 40
CENTRO.CIRCULO 50
FIM
```

CONCEITOS BÁSICOS

"Refogue o tempero." Ao dar essa instrução supondo que a leitora já saiba refogar, a receita lança mão de um algoritmo básico, ao invés de descrever todo um processo. Isso vale também para programas de computação.

Um algoritmo descreve qualquer processo em termos de outros processos anteriormente definidos (como rotinas já escritas no programa) ou tão básicos que dispensem definições (como comandos e funções preexistentes nas linguagens computacionais).

Algoritmos básicos formam outros, mais gerais, que, por sua vez, integram outros maiores, até que todo o programa seja escrito como um só algoritmo, formado por outros de nível inferior. Este é o conceito básico de programação estruturada ou modular.

A matemática, a eletrônica, a estatística e outras ciências têm seus algoritmos, transportáveis para a linguagem computacional.

Todo algoritmo apresenta uma entrada e uma saída, isto é, trabalha com dados iniciais, ou parâmetros, para produzir resultados. Por exemplo: em LEFT$(P$,5), P$ e 5 são os parâmetros passados à função LEFT$. Esta, por sua vez, devolve um resultado em forma de string alfanumérico.

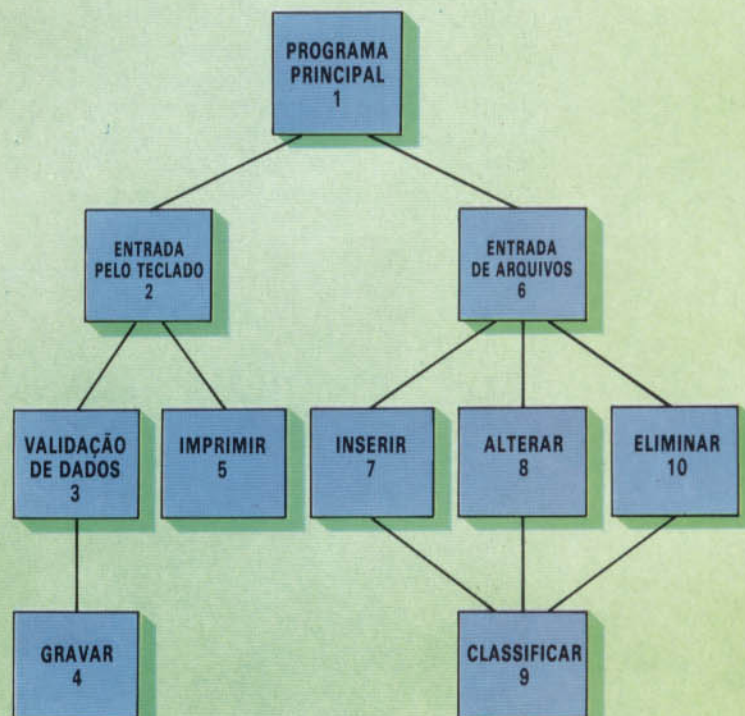
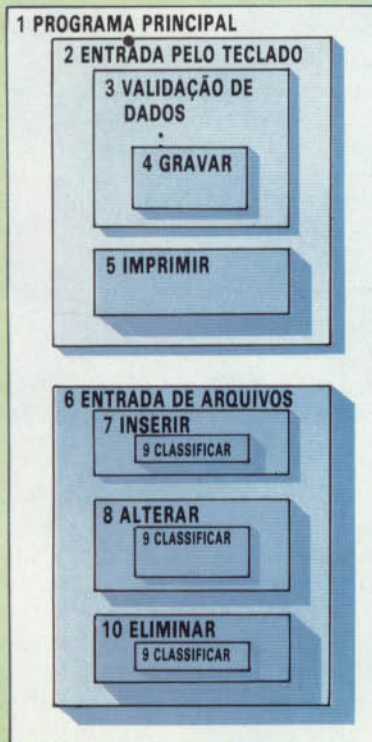
A base da elaboração de um algoritmo, portanto, reside na definição do conteúdo dos parâmetros de entrada e saída; seu tipo (inteiro, com decimais, alfanumérico etc.); e sua faixa de amplitude.

As instruções de um bom algoritmo caracterizam-se pela precisão (ausência de ambigüidades) e pela eficácia (não solicitando tarefas impossíveis, como o cálculo do maior número primo). O algoritmo precisa ser finito — apresentar testes eficientes para condições de saída — e não pode cair num loop sem fim. Suas boas qualidades incluem rapidez de execução e economia de memória. Convém, ainda, que seja simples e engenhoso ("elegante").

Importa bastante, também, a generalidade do algoritmo, sua capacidade de lidar com muitas situações diferentes daquela para a qual foi projetado. Desse modo, o programador monta uma biblioteca de rotinas reutilizáveis, como as que aceitam dados, as que mostram mensagens ao usuário, as que calculam funções etc.

Blocos e estruturas
O diagrama de blocos da esquerda mostra como os algoritmos de um programa se inserem uns nos outros. O fluxograma à direita realça as articulações entre eles e os níveis de hierarquia, no mesmo programa. Os algoritmos mais primitivos são os mais internos e mais baixos na hierarquia.

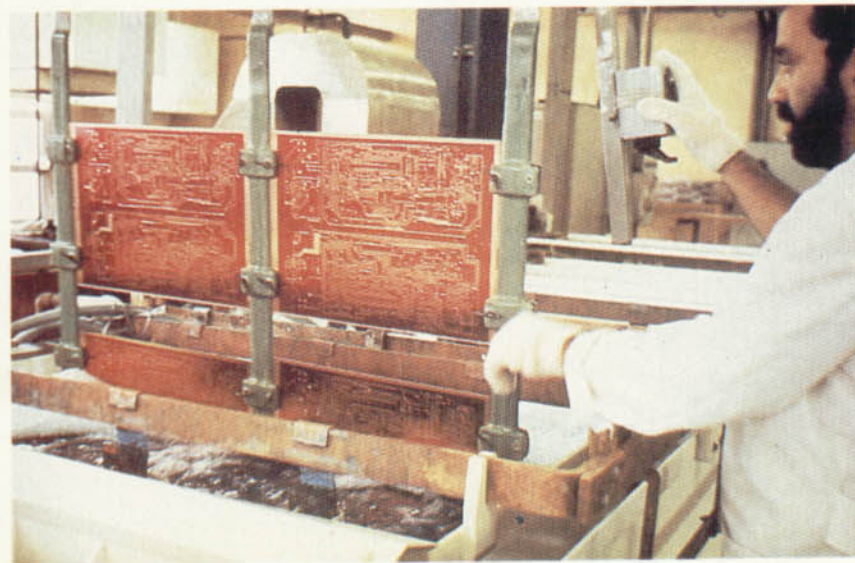
1 PROGRAMA PRINCIPAL
2 ENTRADA PELO TECLADO
3 VALIDAÇÃO DE DADOS
4 GRAVAR
5 IMPRIMIR
6 ENTRADA DE ARQUIVOS
7 INSERIR
9 CLASSIFICAR
8 ALTERAR
9 CLASSIFICAR
10 ELIMINAR
9 CLASSIFICAR

PROGRAMA PRINCIPAL 1
ENTRADA PELO TECLADO 2
ENTRADA DE ARQUIVOS 6
VALIDAÇÃO DE DADOS 3
IMPRIMIR 5
INSERIR 7
ALTERAR 8
ELIMINAR 10
GRAVAR 4
CLASSIFICAR 9

TERMINAL BANCÁRIO

No comando
Carlos Eduardo Correa da Fonseca, diretor-superintendente da Itautec, em 1985.

Voltada inicialmente para a montagem de sistemas bancários automatizados, a Itautec colocou-se entre as cinco maiores empresas nacionais de informática, produzindo a média de 1.500 equipamentos por mês.

Empresa do grupo financeiro Itaú, a Itaú Tecnologia S.A. (Itautec) foi fundada em novembro de 1979, com o projeto inicial de instalar os sistemas de agências *on line* (''em linha'', ou seja, controlados diretamente por um computador central) para o Banco Itaú, de modo a automatizar os serviços bancários. Seguindo o modelo americano do Citibank, que produz toda a rede de equipamentos necessários à montagem do sistema on line, um grupo de engenheiros do Itaú iniciou um trabalho de pesquisa tecnológica para o desenvolvimento do projeto.

Em junho de 1980, já se encontravam instalados cem terminais. Basicamente, o sistema BANKTEC consiste em terminais ligados por linhas telefônicas a um computador central, onde ficam armazenadas as informações relativas a cada cliente. O sistema conta com diversos equipamentos terminais: unidade de resposta audível, terminal caixa, terminal administrativo, terminal operado pelo cliente, terminal extrato operado pelo cliente e terminal transferência de fundos. Entre terminais e concentradores, a Itautec forneceu 7.000 equipamentos para a montagem do sistema on line do Banco Itaú. Essa marca foi conseguida graças à produção de duas fábricas — de circuitos impressos e de equipamentos de eletrônica digital. Fabricando a média de 1.500 unidades/mês, a indústria de equipamentos encontra-se instalada numa área de 8.000 m².

A fábrica de circuitos impressos ocupa área de 2.000 m² e atende tanto às necessidades da empresa como à crescente demanda do mercado de circuitos impressos.

Mediante convênios, a Itautec mantém intercâmbio de equipamentos, tecnologia e know-how com importantes universidades do país.

No final de 1982, a empresa lançou seu primeiro microcomputador, o I-7000 de 8 bits, destinado à aplicação profissional. Na mesma linha de microcomputadores, entraram no mercado versões simplificadas desse equipamento, o I-7000 Jr. e o I-7000 Jr. E — este só tem possibilidade de uma placa de expansão. O I-7000 Jr. é uma versão ainda mais simplificada para baratear os custos do produto. Não possui teclado numérico. Atende às necessidades de processamento de dados da pequena empresa e dos profissionais liberais.

Em novembro de 1984, na Feira de Informática, foi lançado o I-7000 PCxt, de 16 bits, que contou com a experiência anterior, mas tomou por referência os PCs da IBM lançados nos Estados Unidos — o IBM PC e o IBM PCxt. O I-7000 PCxt da Itautec opera tanto com o sistema SIM/M quanto com o sistema SIM/DOS. Ou seja, é compatível com o I-7000 e com os IBM PC e PCxt.

Depois do lançamento do I-7000 PCxt, comercializado a partir de maio de 1985, a Itautec passa a desenvolver na mesma linha de tecnologia o IBM PCat americano, que será o próximo I-7000 a atender ao mercado nacional.

Na Feira de Informática de 1984 foi lançado também o IFAX 3021 — a primeira telecopiadora nacional. Funciona acoplada a telefone e, pelos circuitos de telecomunicação, transmite e reproduz de forma permanente textos, gráficos, ilustrações, desenhos e fotos. Essa transmissão se realiza pela rede nacional ou internacional de telecomunicações.

A Itautec é responsável também pelo lançamento de um dos primeiros computadores de médio porte brasileiros: o I-9000. Trata-se de um sistema completo de processamento de dados de 32 bits, compatível com IBM, o que permite o acesso a um grande conjunto de software disponível no mercado. A primeira unidade produzida foi vendida à Universidade de Campinas (Unicamp), em São Paulo, ainda em 1984.

Outro sistema desenvolvido pela Itautec é o videotexto. A empresa vem produzindo terminais, placas de expansão para microcomputador, concentradores telefônicos, equipamentos para gravação e atualização de dados e sistema de editoração. Com isso, a Itautec passa a ser a única a oferecer a solução completa para o videotexto — um sistema em que o usuário pode selecionar e acessar os mais variados tipos de informação utilizando a rede telefônica pública.

Indústria de componentes
Em seus 2.000 m² de área, a fábrica de circuitos impressos garante a produção para atender à crescente demanda do mercado de componentes.

COMPUTADOR ESPIÃO

Em quase todo o mundo moderno — inclusive no Brasil —, órgãos responsáveis pelo cumprimento da lei envolvem-se em atividades ilegais ao lançar mão dos mais recentes avanços da espionagem eletrônica.

O uso oficial da espionagem computadorizada vem ganhando terreno e atinge cada vez mais nosso cotidiano.

Em certos casos, o controle tornou-se rotina de vital importância para os órgãos de segurança. É o caso do registro de certificados de propriedade de automóvel, carteiras de motorista e carteiras de identidade. A possibilidade de integração dos vários sistemas é crescente, devido à correlação entre os arquivos e, por exemplo, os computadores da polícia. Isso dá às autoridades uma capacidade extra de controle e vigilância das atividades de toda a população.

Em muitos países já existe preocupação com essa possibilidade e iniciativas na elaboração de leis que protejam o cidadão do abuso de poder. No entanto, alguns acreditam que tais leis serão constantemente atropeladas pelos avanços da tecnologia.

Muitos dos processos utilizados em espionagem computadorizada e em sistemas de segurança baseiam-se no reconhecimento de padrões. A técnica permite ao computador comparar aquilo que ele "vê" com modelos já armazenados em sua memória. A desvantagem desse tipo de sistema é que ocupa muito espaço de memória e muita capacidade de processamento, além de um dispêndio de tempo relativamente grande. Mas a disponibilidade e o barateamento do espaço de memória e da capacidade de processamento possibilitaram importantes avanços nessa área.

Policiais auxiliares

Um exemplo desse avanço é o sistema de impressões digitais instalado na década de 80 em alguns países europeus. Ele tem capacidade de armazenar 650.000 impressões digitais e 100.000 "marcas", isto é, impressões parciais encontradas em locais onde ocorreram crimes. O sistema compara as marcas com as impressões armazenadas e verifica se correspondem a alguma delas.

"Grampeando" o escritório
Há inúmeros lugares onde os aparelhos de escuta podem ser escondidos, variando do mais óbvio aos de acesso mais difícil. Na foto, os percevejos poderiam estar ocultos nos seguintes locais:

1 *O telefone.* O percevejo pode estar no bocal, no corpo do telefone ou na luminária ao lado.

2 *O vaso de planta.* Na terra, embaixo do vaso ou camuflado como um percevejo real.

3 *A parede.* Percevejos já foram encontrados enfiados na parede como tachas ou atrás de revestimentos de cortiça.

4 *O quadro.* Disfarçado como um gancho de suporte ou escondido atrás da moldura.

5 *A escrivaninha.* Embaixo do tampo ou em qualquer das gavetas.

As ferramentas do ofício

Proteção computadorizada
Pessoas que lidam com informações sigilosas protegem-se da espionagem eletrônica usando dispositivos como este despistador telefônico computadorizado. Em vez de falar ao telefone, o usuário do aparelho digita sua mensagem no teclado.
O dispositivo a envia de forma silenciosa pelas linhas telefônicas usuais.
A mensagem é então recebida na tela de um sistema complementar. Um sintetizador de voz também pode converter os impulsos que chegam em mensagem audível.

Mensagem em código
Similar ao despistador, esse sistema manda uma mensagem codificada a partir de um manuscrito. A mensagem codificada conserva-se protegida de bisbilhoteiros e "grampeadores", pois é transmitida silenciosamente. Mesmo assinaturas podem ser enviadas dessa forma.

Analisador de tensão
Mede a incidência de tensão na voz. O aparelho demonstra os resultados imediatamente, sob forma numérica simples. Trata-se de sofisticado detector de mentiras e indica também se a pessoa está tensa ou ansiosa.

Essa aplicação é conseguida por minicomputadores capazes de processar matrizes altamente elaboradas, monitores e câmaras de televisão de excelente desempenho. Mesmo assim, o equipamento pode checar apenas duzentas ou trezentas marcas por dia.

Um sistema similar de reconhecimento e comparação de padrões foi montado numa ponte de grande movimento, na Inglaterra. Câmaras dirigidas às pistas de trânsito captam imagens das placas dos carros que se aproximam. O computador analisa as imagens e compara os números das placas com os de um arquivo de carros procurados. Quando é o caso, transmite-se por rádio, diretamente às viaturas da polícia rodoviária, a informação de que um desses carros foi visto.

Uma das áreas em que o desenvolvimento de hardware para microcomputadores teve efeitos mais significativos foi na produção de dispositivos de espionagem. Por suas dimensões cada vez menores, foram apelidados de "percevejos".

A tecnologia do chip possibilitou a produção de radiotransmissores do tamanho de um grão de arroz com sofisticados controles eletrônicos incorporados. Um dispositivo típico dessa categoria extrai sua energia da fonte que alimenta o telefone em que está instalado.

Há também percevejos com fonte de energia própria. Se deixados num canto da sala, captarão toda a conversa daquele ambiente e a transmitirão a um receptor distante. Como o outro, ele só é acionado quando alguém estiver conversando.

Mais ao estilo James Bond, existe ainda o percevejo remoto. O dispositivo lança um raio laser a uma janela e capta as vibrações provocadas no vidro por uma conversa. A conversa é restaurada, por computador, a partir das interferências que aparecem no raio refletido.

Em operações militares, o computador tem ainda outra função. Com o objetivo de evitar a espionagem do inimigo, os transmissores e receptores de rádio, no campo de batalha, utilizam um sistema controlado por microprocessadores. A mensagem sofre modificação de freqüência e "pula" para a outra segundo um código preestabelecido. Utiliza-se esse procedimento para evitar que outro receptor capte a transmissão.

Terminais do futuro

Os computadores usados para espionagem e segurança são equipamentos de grande porte, do mesmo tipo e capacidade das máquinas usadas na decifração de códigos secretos em órgãos como a Agência Central de Informações (CIA), nos Estados Unidos. Mas os avanços da tecnologia de hardware caminham para a miniaturização. Logo a identificação de impressões digitais, vozes e rostos será feita em minutos e a baixo custo.

Num futuro próximo, carros de polícia deverão estar equipados com computadores capazes de acessar instantaneamente dados de um registro escolar, criminal, médico, de seguro social ou qualquer outro arquivo oficial, pela simples introdução de um cartão magnético de identificação numa das entradas do terminal. A máquina analisará impressões digitais ou fotografias comparando-as com as dos arquivos centrais para certificar-se da identidade do suspeito.

DIVIDA E GOVERNE

Valendo-se de procedimentos recursivos (que chamam a si próprios), o LOGO produz imagens gráficas de características incomuns inventadas por matemáticos para ilustrar certos paradoxos.

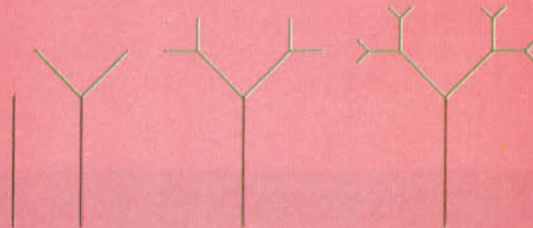
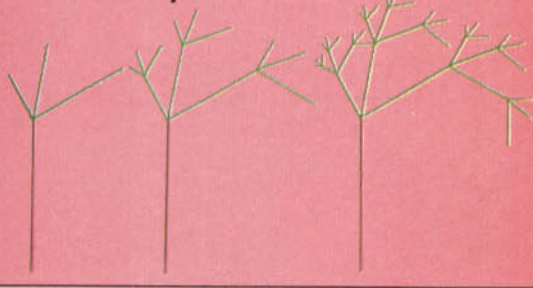
Ilustra a recursion no LOGO um programa projetado para traçar figuras de "árvores". No princípio, desenhamos apenas o tronco com dois galhos, um à direita e outro à esquerda. A ramagem é produzida do mesmo modo (embora seus galhos sejam cada vez menores), com um caule central e os galhos direito e esquerdo.

O procedimento para o traçado de uma "árvore binária" desse tipo requer a entrada de dois dados: o comprimento do caule e o grau do "nível". O comprimento dos ramos divide-se ao meio a cada nível que se afasta do tronco.

```
AP RAMO :COMP :NIVEL
  SE :NIVEL = 0 ENTAO PARE
  FR :COMP
  ES 45
  RAMO (:COMP/2) (:NIVEL - 1)
  DI 90
  RAMO (:COMP/2) (:NIVEL - 1)
  ES 45
  VO :COMP
FIM
```

Observe que o procedimento não modifica a posição e a direção — ou seja, o "estado" — da tartaruga. Isso é importante, pois, de outro modo, o estado dela mudaria todas as vezes que o procedimento chamasse a si mesmo, tornando impossível a realização do desenho.

A árvore traçada por esse procedimento não é das mais realistas. Para torná-la mais parecida com as árvores verdadeiras, pode-se adaptar de várias maneiras o procedimento. A versão que apresentamos em seguida traça, a cada nível, três ramos de comprimento diferente:

```
AP RAMO1 :COMP :NIVEL
  SE :NIVEL = 0 ENTAO PARE
  FR :COMP
  ES 30
  RAMO1 (:COMP/3) (:NIVEL - 1)
  DI 40
  RAMO1 (:COMP/2) (:NIVEL - 1)
  DI 50
  RAMO1 (:COMP/1.5) (:NIVEL - 1)
  ES 60
  VO :COMP
FIM
```

Polígonos quadriculados

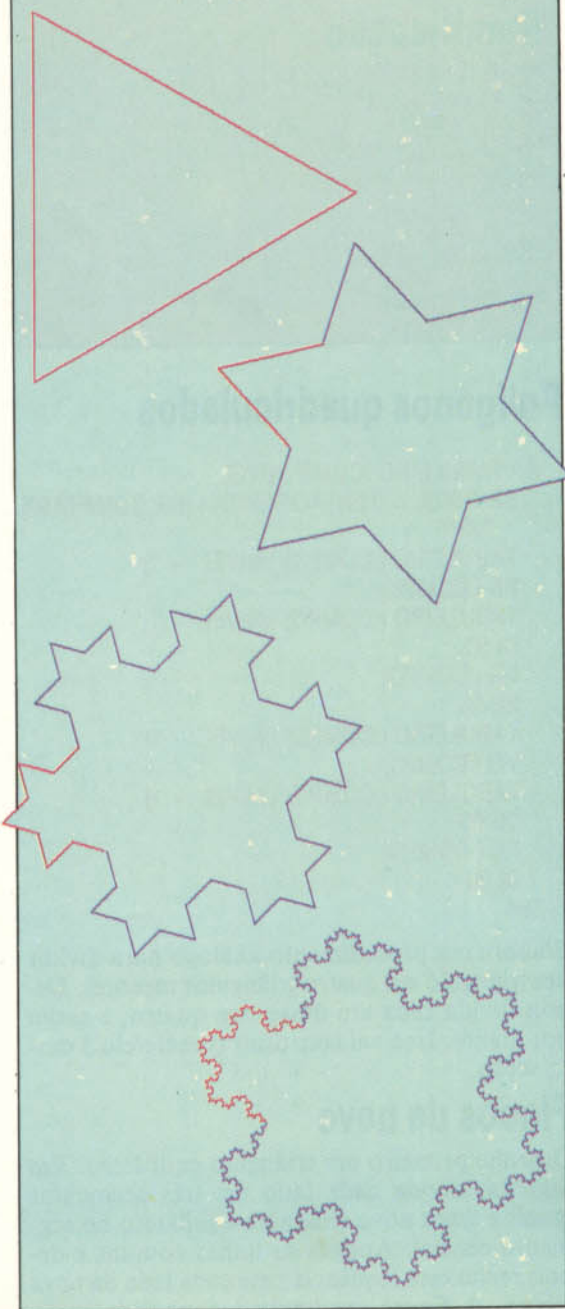
```
AP TABULEIRO :COMP :NIVEL
  SE :NIVEL = 0 ENTAO REPITA 4 [FR :COMP DI 90]
    PARE
  TABULEIRO (:COMP/2) (:NIVEL - 1)
  FR (:COMP/2)
  TABULEIRO (:COMP/2) (:NIVEL - 1)
  DI 90
  FR (:COMP/2)
  ES 90
  TABULEIRO (:COMP/2) (:NIVEL - 1)
  VO (:COMP/2)
  TABULEIRO (:COMP/2) (:NIVEL - 1)
  ES 90
  FR (:COMP/2)
  DI 90
FIM
```

Elabore um procedimento análogo para dividir um triângulo em quatro triângulos menores. Depois divida cada um desses em quatro, e assim por diante. Isso vai constituir o **exercício 5** desta seção.

Flocos de neve

Desenhe primeiro um triângulo eqüilátero. Em seguida, divida cada lado em três segmentos iguais e trace novo triângulo eqüilátero no segmento central. Apague as linhas comuns e depois repita essa seqüência para cada lado da nova figura. A figura resultante é conhecida como

"curva dos flocos de neve", por causa de sua aparência.

```
AP NEVE :TAM :NIVEL
  REPITA 3 (LADO :TAM :NIVEL DI 120)
FIM
AP LADO :TAM :NIVEL
  SE :NIVEL = 0 ENTAO FR :TAM PARE
  LADO (:TAM/3) (:NIVEL - 1)
  ES 60
  LADO (:TAM/3) (:NIVEL - 1)
  DI 120
  LADO (:TAM/3) (:NIVEL - 1)
  ES 60
  LADO (:TAM/3) (:NIVEL - 1)
FIM
```

Note que LADO não é transparente ao estado, mas deixa a tartaruga no ponto certo para o traçado da figura seguinte.

Se esse processo de divisão se repetir indefinidamente (até o limite, na expressão dos matemáticos), o resultado será uma curva de comprimento infinito que, no entanto, circunda uma área finita. Pode-se provar que essa curva não é unidimensional nem bidimensional, mas possui características intermediárias.

Linha semelhante é montada a partir de um quadrado, dividindo-se cada um de seus lados em três partes iguais para construir quadrados nos segmentos médios, e assim por diante. Tente elaborar um procedimento que faça isso. **(Exercício 6.)**

As curvas mostradas a seguir foram inventadas pelo matemático polonês Waclaw Sierpinski (1882-1969). Se o processo é levado até o limite, obtém-se uma curva (uma linha unidimensional) que passa por todos os pontos do quadrado circundante (uma figura bidimensional). Várias outras curvas desse tipo apresentam comprimento pouco usual.

O procedimento empregado para o traçado delas é complexo. No nível 1, por exemplo, a curva compõe-se de quatro lados (mostrados em azul), interligados por quatro diagonais.

Assim, o procedimento principal, SIERP, apenas divide o processo em quatro seções, para que o procedimento UM.LADO opere com um deles por vez.

Considere agora apenas um dos lados, que é formado de três linhas — uma diagonal, uma vertical ou horizontal e outra diagonal. No nível 2, um conjunto menor de três linhas substitui cada uma das diagonais. A linha vertical ou horizontal é substituída por dois conjuntos semelhantes de três linhas unidas por outra.

A seguir apresentamos os procedimentos para o traçado das curvas, com o comando FACA usado para inicializar DIAG:

```
AP SIERP :LADO :NIVEL
  FACA "DIAG :LADO / RQD2
  REPITA 4 [UM.LADO :NIVEL DI 45 FR :DIAG DI 45]
FIM

AP UM.LADO :NIVEL
  SE :NIVEL = 0 PARE
UM.LADO (:NIVEL - 1)
  DI 45
  FR :DIAG
  DI 45
  UM.LADO (:NIVEL - 1)
  ES 90
  FR :LADO
  ES 90
  UM.LADO (:NIVEL - 1)
  DI 45
  FR :DIAG
  DI 45
  UM.LADO (:NIVEL - 1)
FIM
```

Curva de Sierpinski

Respostas do exercício 4

Um procedimento recorrente para o traçado da torre de quadrados:

```
AP TORRE :TAM
  SE :TAM < 5 ENTAO PARE
  QUAD :TAM
  MOVA :TAM
  TORRE (:TAM/2)
FIM

AP QUAD :TAM
  REPITA 4 [FR :TAM DI 90]
FIM

AP MOVA :TAM
  FR :TAM
  DI 90
  FR (:TAM/4)
  ES 90
FIM
```

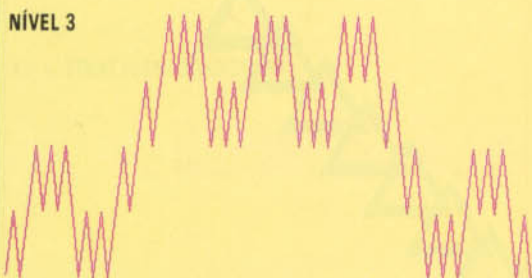
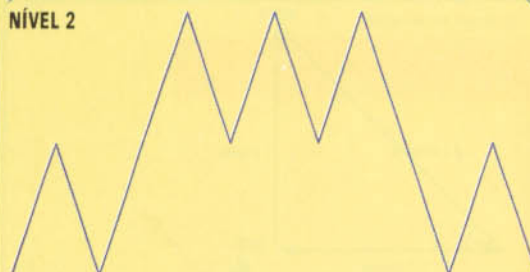
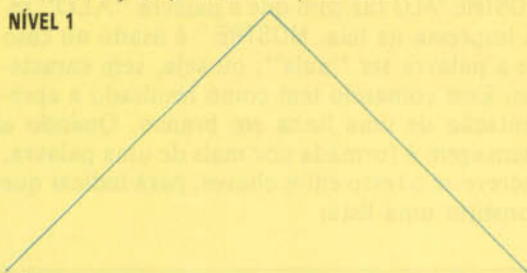
Simplificação

Abreviações dos comandos apresentados nesta matéria:

FRENTE	FR
VOLTE	VO
DIREITA	DI
ESQUERDA	ES
LISTAR	LI

Outros comandos empregados:
SE
ENTAO
REPITA
FACA
FIM

Exercício 7

As ilustrações abaixo mostram figuras que, no limite, definem curvas sem gradiente. O primeiro nível consiste em duas linhas — uma que sobe e outra que desce. Para passar ao nível seguinte, substitua a linha ascendente por uma linha interrompida formada de seis partes. Esta se eleva até a metade da altura da linha original e, em seguida, desce até embaixo. Depois, sobe de novo até a metade da altura, continua até a altura máxima, cai até a metade e, por fim, sobe de novo até a altura máxima. A linha descendente divide-se em seis seções pelo mesmo processo.

Tente escrever um conjunto de procedimentos que trace essa série de curvas. Use o comando DEFXY em vez de FR e DI. Seu procedimento mais abrangente deve dividir a tarefa em duas partes — uma para a subida e outra para a descida. A seguir, serão necessários dois procedimentos separados para cada parte do procedimento de nível mais alto. (Tais procedimentos podem chamar-se uns aos outros, assim como a si mesmos.)

PERDIDA NO ESPAÇO

A tartaruga do LOGO perdeu-se nos confins do universo e precisa de sua ajuda para encontrar o caminho de volta. Antes de criarmos este jogo, porém, veremos alguns métodos de entrada e de saída de dados.

No jogo Tartaruga no Espaço, a tartaruga está perdida nos confins do universo, a enorme distância de sua base, para a qual deve retornar. O jogo requer a apresentação de diversas mensagens na tela e o comando para implementá-las é MOSTRE. Assim que a mensagem é mostrada na tela, o cursor se transfere para o início da linha seguinte.

Nas mensagens de apenas uma palavra, basta digitar a palavra em seguida a MOSTRE — assim, MOSTRE "ALO faz com que a palavra "ALO" seja impressa na tela. MOSTRE " é usado no caso de a palavra ser "nula", ou seja, sem caracteres. Esse comando tem como resultado a apresentação de uma linha em branco. Quando a mensagem é formada por mais de uma palavra, escreve-se o texto entre chaves, para indicar que constitui uma lista:

```
MOSTRE [SEU TEMPO TERMINOU]
```

MOSTRE também é usado para se apresentar o conteúdo de uma variável. Desse modo, MOSTRE :RESULTADO buscará o valor associado à variável "RESULTADO e o mostrará na tela. Mensagens e valores variáveis podem ser combinados na mesma declaração MOSTRE, colocando-se a instrução completa entre parênteses. Por exemplo:

```
(MOSTRE [SEU RESULTADO FOI] :RESULTADO)
```

MOSTRAR funciona do mesmo modo que MOSTRE. A única diferença entre os dois comandos é que em MOSTRAR o cursor permanecerá no final do texto impresso e não será transferido para a linha seguinte, como em MOSTRE. Para verificar isso, digite:

```
MOSTRAR [QUAL E O SEU NOME]
```

Operações de saída

Os comandos LOGO do tipo SEMT ou MOSTRET fazem com que algo aconteça, isto é, produzem um efeito na tartaruga. No entanto, outros primitivos LOGO — CORX, por exemplo — apresentam valores, em vez de produzir efeitos. Em geral, esses valores são usados como dados de entrada para os comandos. Assim, por exemplo, a digitação de:

```
MOSTRE CORX
```

faz com que CORX transfira o valor correspondente à coordenada x da tartaruga para o comando MOSTRE, o qual apresenta o resultado na tela. Portanto, se em determinado momento o valor de CORX for 20, MOSTRE CORX fará com que o número 20 apareça na tela. Se apenas CORX for digitado, a mensagem IGUAL: 20 será mostrada. Na verdade, essa é uma mensagem de erro.

Todos os procedimentos que vimos até agora eram comandos. No caso das *operações,* utilizaremos o primitivo SAIDA. Como exemplo, veremos um procedimento que tem como resultado a distância percorrida pela tartaruga desde sua posição inicial. Nesse procedimento, RQD calcula a raiz quadrada de um número:

```
AP DISTANCIA
  SAIDA RQD (CORX*CORY + CORY*CORX)
FIM
```

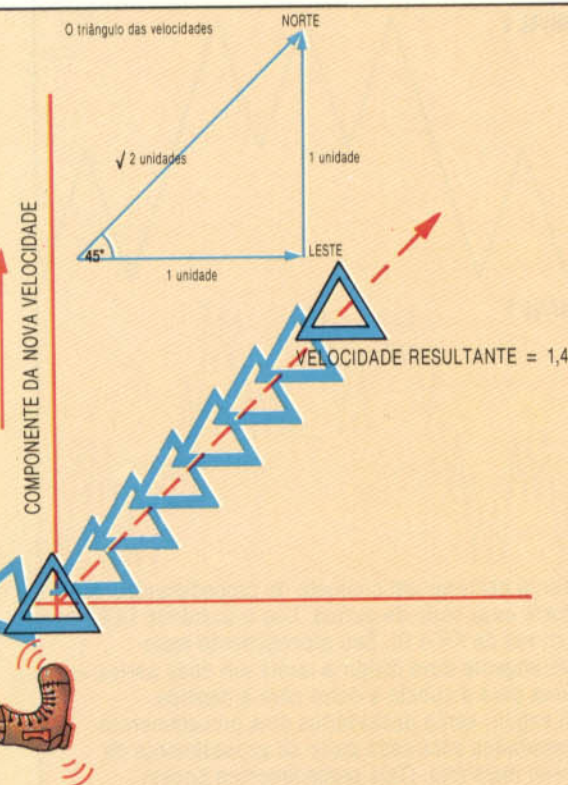
Tente movimentar a tartaruga para diferentes pontos da tela. Use DISTANCIA para determinar o quão longe ela está de sua origem. Por exemplo, DEFXY 30 40 MOSTRE DISTANCIA deve ter como resultado 50.

Quando o LOGO executa uma instrução SAIDA, interrompe o procedimento que está executan-

O efeito do chute

Com a velocidade inicial de 1, a tartaruga parte para o leste. Em seguida, é direcionada para o norte e leva um "chute". Assim, duas velocidades passam a atuar sobre ela. Com um ângulo reto entre si, essas velocidades formam os lados de um triângulo retângulo, cuja hipotenusa representa a magnitude e a direção da velocidade resultante. Por meio do teorema de Pitágoras, obtemos o valor de 1,414 (a raiz quadrada de 2) para a velocidade resultante. Como esse triângulo é isósceles, a tartaruga passa a movimentar-se na direção nordeste. Observe, contudo, que ela continua orientada para o norte.

O triângulo das velocidades
NORTE
√2 unidades
1 unidade
45°
1 unidade
LESTE
COMPONENTE DA NOVA VELOCIDADE
VELOCIDADE RESULTANTE = 1,4
VELOCIDADE INICIAL
CHUTE

KEVIN JONES

do no momento e passa o controle ao procedimento que o chamou. Isso pode ser verificado no procedimento MAX, que apresenta como dado de saída o maior dentre dois números:

```
AP MAX :X :Y
  SE :X > :Y ENTAO SAIDA :X
  SAIDA :Y
FIM
```

MOSTRE MAX 6 2 apresentará como resultado o número 6. Tente criar um procedimento que dê o valor absoluto de um número, de modo que tanto MOSTRE ABS 4 como MOSTRE ABS (– 4) terão como resultado o valor 4.

O jogo Tartaruga no Espaço pedirá que você digite seu nome e pressione RETURN. Isso é feito pelo procedimento que mostramos a seguir:

```
AP DAR.NOME
  MEIATELA
  MOSTRAR [QUAL E O SEU NOME?]
  FACA "NOME PRIMEIRO ENTRE
  (MOSTRAR "ALO :NOME)
FIM
```

ENTRE espera que uma linha seja digitada e finalizada com um RETURN. Ele então apresenta a linha sob a forma de lista. PRIMEIRO produz o primeiro elemento de uma lista. Experimente o procedimento DAR.NOME e digite o nome "PAULO". Em seguida, veja o que acontece ao digitar "PAULO MARCELO".

Os movimentos da tartaruga na tela serão controlados pelo acionamento das teclas D, E e C. D fará com que a tartaruga gire 30° em sentido horário (para a direita); E fará com que ela gire 30° em sentido anti-horário (para a esquerda). Por fim, C é usado para "chutar" a tartaruga — aumentando a velocidade dela qualquer que seja sua direção em determinado momento. A tartaruga estará em movimento pela tela e reagirá de imediato se as teclas forem pressionadas. Seria muito útil se dispuséssemos de um primitivo LOGO — LEIATECLA, talvez — que mostrasse a última tecla acionada. Se ele existisse, poderíamos escrever:

```
AP COMANDO
  FACA "COM LEIATECLA
  SE :COM = "D ENTAO DI 30
  SE :COM = "E ENTAO ES 30
  SE :COM = "C ENTAO CHUTE
FIM
```

Não existe um primitivo desse tipo. Contudo, podemos escrevê-lo sob a forma de procedimento:

```
AP LEIATECLA
  SE CP? ENTAO SAIDA PEGUE
  SAIDA"
FIM
```

Quando se pressiona uma tecla, ela é gravada na memória temporária (buffer) do teclado. PEGUE apenas lê o último caractere introduzido no buffer. Caso este esteja vazio, PEGUE esperará que uma tecla seja pressionada para ler o caractere relevante. CP? será verdadeiro se o buffer contiver algum caractere, e falso se estiver vazio. Desse modo, LEIATECLA terá como resultado o último caractere introduzido no buffer, ou então uma palavra nula no caso de este estar vazio.

A tartaruga dinâmica

Na realidade, essa tartaruga viajando pelo espaço é uma "tartaruga dinâmica": tem velocidade, assim como posição e direção, como qualquer tartaruga viva.

Por estar no espaço, não haverá atrito ou gravidade. Ela obedecerá, portanto, às leis do movimento formuladas por Newton. Nossa ilustração tornará isso mais claro, mas, como exemplo, vamos supor que a tartaruga esteja se movendo da esquerda para a direita da tela com velocidade de 1 unidade. Se a tecla E for pressionada, a tartaruga mudará de direção, passando a mover-se para a parte de cima da tela. No entanto, seu impulso fará com que continue se movimentando na horizontal. Quando a tecla C é acionada, a tartaruga leva um "chute" na direção em que foi posicionada. Se for impulsionada para cima com a velocidade 1, o movimento resultante terá a velocidade de 1,4 na diagonal. Portanto, a tartaruga dinâmica permite a realização de experiências com um corpo sujeito às leis de Newton. Ela foi especialmente projetada para que você desenvolva uma compreensão intuitiva dessas leis, sem que seja necessário entender todas as equações envolvidas.

No programa, a velocidade da tartaruga dinâmica é considerada em termos de duas componentes, relativas aos eixos x e y. Essas componentes são determinadas por meio das funções SEN e COS. Os únicos controles são os três já mencionados. Para começar o jogo, digite INICIO. Você dispõe de tempo limitado para alcançar seu objetivo, e o programa mantém um registro do melhor resultado obtido.

Tartaruga extraterrestre
O programa que vimos desenha apenas a tartaruga e seu alvo. As estrelas e os planetas destas figuras foram criados por simples procedimentos para o traçado de círculos.

Projeto de programa

Elabore um programa para o jogo Alunissagem. Nele, você está pilotando uma nave a certa distância acima da superfície lunar. Resta pouco combustível e, devido à atração constante da gravidade, a velocidade de descida também aumenta de forma constante a cada segundo. Acionando a tecla F, você liga os motores da nave, reduzindo a velocidade de descida. Isso provoca, no entanto, maior consumo de combustível.
O objetivo do jogo é fazer com que a descida seja lenta o suficiente para permitir uma alunissagem segura.

Novos comandos

CORX
CORY
DEFXY
MEIATELA
TODATELA

Respostas do exercício 5

A) Triângulos aninhados

```
AP TRI :TAM :NIVEL
  SE :NIVEL = 0 ENTAO REPITA
3 [FR :TAM DI 120]
    PARE
  TRI (:TAM/2) (:NIVEL - 1)
  FR (:TAM/2)
  TRI (:TAM/2) (:NIVEL - )
  DI 60
  TRI (:TAM/2) (:NIVEL - 1)
  FR (:TAM/2)
  DI 60
  TRI (:TAM/2) (:NIVEL - 1)
  ES 60
  VO (:TAM/2)
  ES 60
  VO (:TAM/2)
FIM
```

B) Floco de neve

```
AP NEVE1 :TAM :NIVEL
  REPITA 4 [LADO1 :TAM :NIVEL DI 90]
FIM
AP LADO1 :TAM :NIVEL
  SE :NIVEL = 0 ENTAO FR :TAM PARE
  LADO1 (:TAM/3) (:NIVEL - 1)
  ES 90
  LADO1 (:TAM/3) (:NIVEL - 1)
  DI 90
  LADO1 (:TAM/3) (:NIVEL - 1)
  DI 90
  LADO1 (:TAM/3) (:NIVEL - 1)
  ES 90
  LADO1 (:TAM/3) (:NIVEL - 1)
FIM
```

C) Curva sem gradiente

```
AP CURVA :FASEX :FASEY :NIVEL
  SOBE :FASEX :FASEY :NIVEL
  DESCE :FASEX :FASEY :NIVEL
FIM
AP SOBE :FASEX :FASEY :NIVEL
  SE :NIVEL = 0 ENTAO DEFXY (CORX + :FASEX)
    (CORY + :FASEY) PARE
  SOBE (:FASEX/6) (:FASEY/2) (:NIVEL - 1)
  DESCE (:FASEX/6) (:FASEY/2) (:NIVEL - 1)
  SOBE (:FASEX/6) (:FASEY/2) (:NIVEL - 1)
  SOBE (:FASEX/6) (:FASEY/2) (:NIVEL - 1)
  DESCE (:FASEX/6) (:FASEY/2) (:NIVEL - 1)
  SOBE (:FASEX/6) (:FASEY/2) (:NIVEL - 1)
FIM

AP DESCE :FASEX :FASEY :NIVEL
  SE :NIVEL = 0 ENTAO DEFXY (CORX + :FASEX)
    (CORY - :FASEY) PARE
  DESCE (:FASEX/6) (:FASEY/2) (:NIVEL - 1)
  SOBE (:FASEX/6) (:FASEY/2) (:NIVEL - 1)
  DESCE (:FASEX/6) (:FASEY/2) (:NIVEL - 1)
  DESCE (:FASEX/6) (:FASEY/2) (:NIVEL - 1)
  SOBE (:FASEX/6) (:FASEY/2) (:NIVEL - 1)
  DESCE (:FASEX/6) (:FASEY/2) (:NIVEL - 1)
FIM
```

Tartaruga no espaço

```
AP INICIO
  FACA "MAX 0
  DESENHE
  ST
  ALVO
  JOGAR
FIM

AP ALVO
  SC DEFXY 0 5 CL
  DI 90
  REPITA 36 [FR 31.4/36
    DI 10]
  SC
FIM

AP JOGAR
  DAR.NOME
  INICIALIZAR
  IMPULSO
FIM

AP DAR.NOME
  MEIATELA
  MOSTRE [QUAL E'O SEU NOME?]
  FACA "NOME PRIMEIRO ENTRE
FIM

AP INICIALIZAR
  DEFXY 100 100
  DEFD 270
  FACA "VELX 0
  FACA "VELY 0
  TODATELA
  MT
FIM

AP IMPULSO
  COMANDO
  DINA.MOVE
  SE DISTANCIA < 10 ENTAO
    FINAL PARE
  FACA RESULTADO :RESULTADO-1
  SE RESULTADO = 0 ENTAO
    TEMPO.ENCERRADO
  PARE
  IMPULSO
FIM

AP COMANDO
  FACA "COM PEGUE
  SE :COM = "D ENTAO DI 30
  SE :COM = "E ENTAO ES 30
  SE :COM = "C ENTAO CHUTE
FIM

AP CHUTE
  FACA "VELX :VELX + 3 * SEN DIRECAO
  FACA "VELY :VELY + 3 * COS DIRECAO
FIM

AP DINA.MOVE
  DEFXY ( CORX + :VELX )
  (CORY + :VELY )
FIM

AP DISTANCIA
  SAIDA RQD
  (CORX * CORX + CORY * CORY )
FIM

AP FINAL
  MOSTRAR"
  MEIATELA
  (MOSTRE [MUITO BEM] :NOME )
  (MOSTRE [SEU RESULTADO FOI]
  :RESULTADO
  REPORTAR
  DE NOVO
FIM

AP REPORTAR
  SE:RESULTADO > :MAX ENTAO FACA
  "MAX :RESULTADO FACA "MELHOR :NOME
  MOSTRAR"
  ( MOSTRE [MELHOR RESULTADO E' ]
  :MELHOR [COM] : MAX [PONTOS] )
FIM

AP DE.NOVO
  MOSTRE [OUTRA PARTIDA?]
  FACA RES PRIMEIRO ENTRE
  SE:RES ="SIM ENTAO NOVO JOGO
  PARE
  SE :RES ="NAO ENTAO PARE
  MOSTRAR [DECIDA-SE, SIM OU NAO?]
  DE.NOVO
FIM

AP TEMPO.ENCERRADO
  MOSTRAR"
  MEIATELA
  MOSTRE [SEU TEMPO TERMINOU]
  DE.NOVO
FIM

AP NOVO.JOGO
  SEMT
  DAR.NOME
  INICIALIZAR
  IMPULSO
FIM
```

ORDENE OS DÍGITOS

No jogo conhecido como Inverso, o objetivo é colocar uma lista de números em ordem crescente, com um mínimo de tentativas. Parece simples, mas os melhores quebra-cabeças derivam de conceitos básicos.

O programa gera, aleatoriamente, uma lista de números para a classificação, e você só muda a ordem dos números invertendo *grupos* específicos de números dentro da lista. Por exemplo, ao pedido de uma lista de nove números, o computador gera esta lista:

2 8 4 7 1 5 6 9 3

O jogador especifica "5" em resposta à mensagem "Inverter?". A ordem dos cinco primeiros números se altera e a lista fica assim:

1 7 4 8 2 5 6 9 3

Aparentemente, uma charada como essa não exige tempo ou esforço para ser resolvida: basta recorrer a um algoritmo preestabelecido. Na prática, porém, é difícil encontrar um algoritmo satisfatório. Suponhamos que haja *n* números na lista. O algoritmo mais óbvio seria este:

- Achar o maior número da lista e inverter todos os números até sua posição. (O número mais alto está agora na extremidade esquerda da lista.)
- Inverter todos os números, de forma que o maior fique na posição desejada (na extremidade direita da lista).
- Encontrar o segundo maior número, e repetir o procedimento. Para levar esse número até a posição desejada, é preciso executar o movimento "Inverter $n-1$".
- Repetir o procedimento até colocar todos os números na ordem.

Esse algoritmo sempre resolve o problema em $2n-3$ movimentos. Mas é possível obter uma solução com menos jogadas.

Para demonstrar como uma estratégia de pensamento avançado pode reduzir o número de jogadas, considere o exemplo do quadro.

Pelo procedimento inicial, gastaríamos sete ($2 \times 5-3$) jogadas, mas um jogador habilidoso resolve o problema em apenas quatro.

Esse programa é um exemplo simples de toda uma série de jogos de inversão. Você pode desenvolver jogos com inversões a partir de qualquer extremidade da lista, ou substituir os números por quadrados de diversas cores.

Quantos números?

```
5   REM *** PARA A LINHA SINCLAIR ***
10  REM ********** INVERSO **********
20  DIM A(20)
30  CLS
35  PRINT "INVERSO"
40  INPUT "QUANTOS NUMEROS ?";N
50  IF N < 0 OR N > 20 OR N < > INT (N) THEN GOTO 30
60  REM *** ORDENANDO A LISTA ***
70  FOR I = 1 TO N
72  LET A(I) = I
74  NEXT I
80  REM *** ESCOLHENDO AO ACASO ***
90  FOR I = 1 TO N
100 LET R = INT ( RND * N + 1)
110 LET X = A(R)
112 LET A(R) = A(I)
114 LET A(I) = X
120 NEXT I
130 T = 1
135 REM *** IMPRIMINDO O QUADRO ***
140 CLS
142 PRINT "INVERSAO NUMERO ";T;" A LISTA E' "
144 PRINT
150 FOR I = 1 TO N
151 PRINT A (I); " ";
152 NEXT I
153 REM *** CHECA SE GANHOU ***
154 LET I = 1
156 IF A(I) = I THEN LET I = I + 1
157 IF I < = N THEN GOTO 156
158 IF I > N THEN GOTO 230
159 REM *** LE A TENTATIVA ***
160 PRINT
162 PRINT
164 INPUT "INVERSAO ? ";R
170 IF R < > INT (R) OR R < 0 OR R > N THEN
    goto 140
180 LET T = T + 1
190 FOR I = 1 TO INT (R / 2)
200 LET X = A(I)
202 LET A(I) = A(R - I + 1)
204 LET A(R - I + 1) = X
210 NEXT I
220 GOTO 140
230 REM *** O VENCEDOR ***
240 PRINT
242 PRINT
244 PRINT "VOCE CONSEGUIU EM "; T ; "TENTATIVAS "
250 PRINT
252 INPUT "QUER JOGAR DE NOVO (S/N)? "; A$
260 IF A$ = "S" THEN RUN
270 CLS
275 END
```

Variações

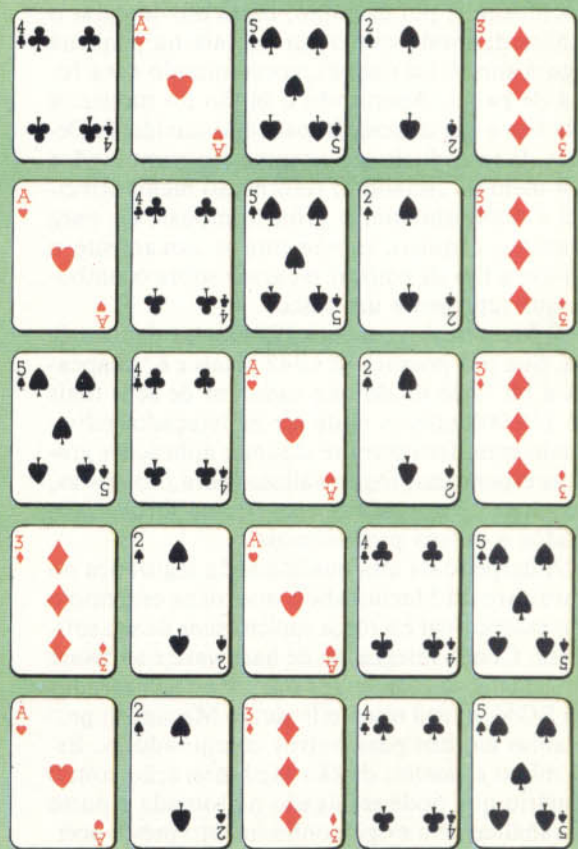
O programa aqui apresentado para a linha Sinclair pode ser adaptado com facilidade para micros de outras linhas.

Os compatíveis com Apple exigem uma única modificação importante, que se refere à sintaxe do comando RND. Assim, na linha 100, escreva:

R = INT (RND (3) * N) 1.

Substitua também CLS por HOME nas linhas 30, 140 e 270.

2-3-5-3

O objetivo de Inverso é classificar uma lista reordenando repetidamente setores dela. O setor a ser invertido deve começar sempre pelo número mais à esquerda, e é descrito pela quantidade de elementos incluídos. Aqui, a seqüência mais satisfatória de números é 2-3-5-3, isto é: inverta as duas primeiras cartas da esquerda; depois, as três primeiras cartas da esquerda; e assim por diante.

MACINTOSH

Combinando a utilização do mouse com menus iconográficos, o Macintosh, da Apple, facilita a interação do usuário com a máquina e garante a seu fabricante a reputação de inovador no mercado de micros.

O Macintosh, nome emprestado de uma seleta variedade de maçã, não se parece com nenhum outro micro do mercado. A começar por seu estojo com alça, bastante pequeno para um equipamento com sua potência de processamento, o que permite classificá-lo como micro portátil. Incluindo o teclado, o mouse e o estojo — com compartimentos para todos os componentes —, o conjunto pesa 11,6 kg. O teclado, semelhante aos das máquinas de escrever, é dotado de processador para manipular funções especiais e conjuntos internacionais de caracteres.

O mais revolucionário no Mac, como costuma ser chamado, é a combinação do mouse que equipa a máquina com os menus iconográficos oferecidos tanto nos programas aplicativos como nos utilitários. Em outras palavras, é o sistema de mais fácil interação com o usuário, uma vez que não exige conhecimento de comandos de operação específicos. Para abrir um arquivo de documentos, por exemplo, basta movimentar o mouse de modo que o cursor caia na pequena figura simbólica (ícone) representando uma folha de papel. Apertando o botão do mouse, a tela fica a sua disposição para essa atividade. Depois de introduzir seu arquivo no computador por meio do teclado, o retorno ao menu principal é realizado com o próprio mouse; aí, para gravar o arquivo, movimenta-se novamente o mouse a fim de colocar o cursor sobre o símbolo que representa um disco.

O Macintosh vem com 128 Kbytes de memória. Sua tela possui 512 x 342 pixels e é ''mapeada a bit'', de modo que cada um de seus mais de 175.000 pontos pode ser endereçado individualmente. Isso permite algumas aplicações gráficas espantosas, muito valiosas para projetistas, arquitetos, consultores, relações-públicas, fotógrafos e outros profissionais.

A despeito da alta qualidade da segurança do hardware do Macintosh, o que torna este micro tão excepcional é a força suplementar de seu software. Com a integração de hardware e software e os intensivos comandos operacionais baseados na ROM, é fácil transferir para o Macintosh programas escritos para outros computadores. Este micro é, assim, de tão fácil interação com o usuário que pode ser ligado na tomada e posto a trabalhar sem exigir conhecimento prévio acerca da operação do computador.

Conexão analógica digital
Podem ser conectados dispositivos analógicos, digitais.

Bus serial
Também conhecido por "slots virtuais", o bus serial permite adicionar toda uma variedade de dispositivos periféricos.

Saída de áudio

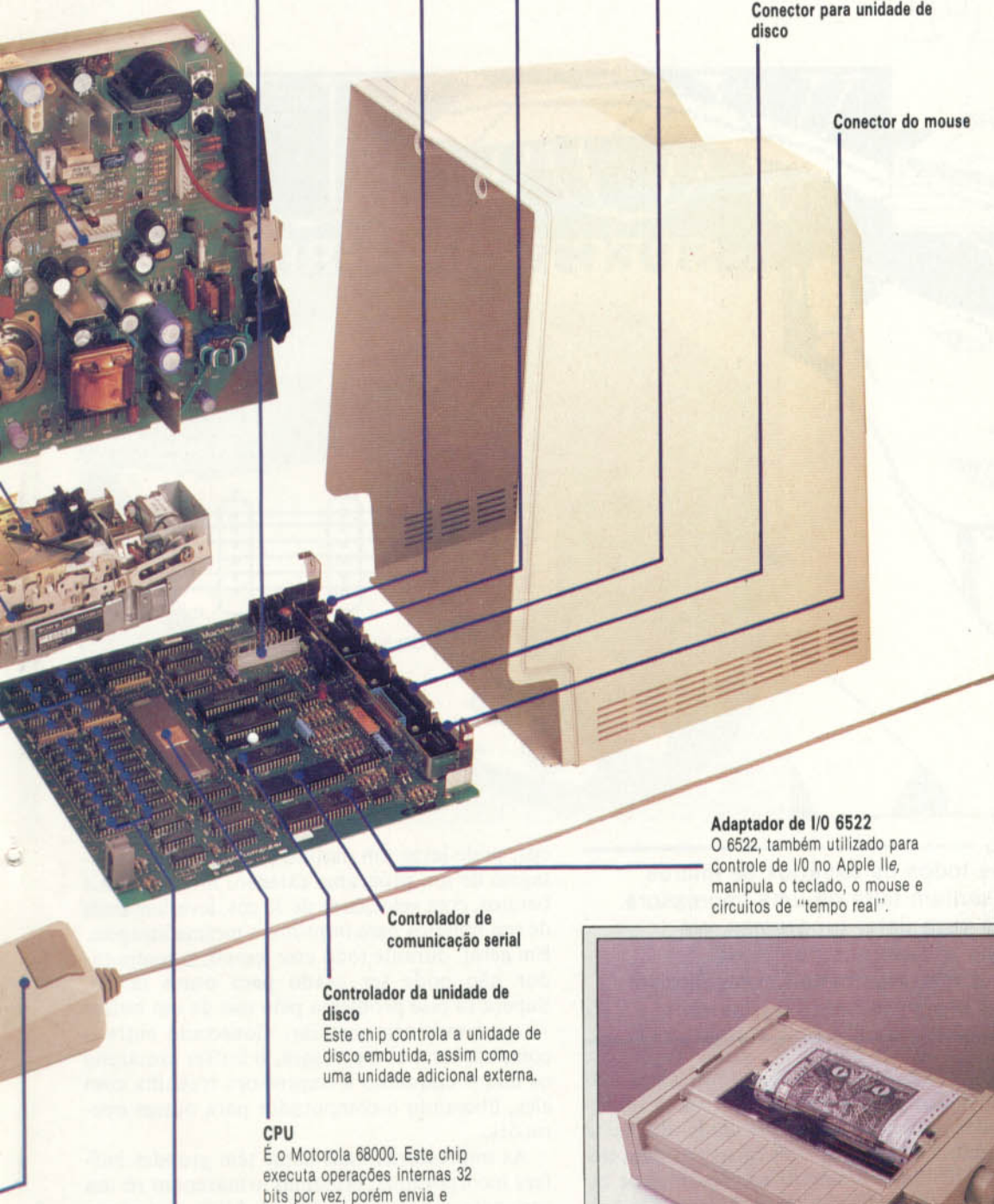
Saída para impressora

Conector para unidade de disco

Conector do mouse

Adaptador de I/0 6522
O 6522, também utilizado para controle de I/0 no Apple II*e*, manipula o teclado, o mouse e circuitos de "tempo real".

Controlador de comunicação serial

Controlador da unidade de disco
Este chip controla a unidade de disco embutida, assim como uma unidade adicional externa.

CPU
É o Motorola 68000. Este chip executa operações internas 32 bits por vez, porém envia e recebe dados a 16 bits; isso representa maior velocidade de processamento e extensão de memória endereçável.

RAM disponível para o usuário, de 128 K
Estes dezesseis chips podem ser substituídos por chips de RAM de 256 K, o que dá ao Macintosh um total de 512 K de memória disponível. No entanto, para as aplicações existentes, bastam 128 K.

MACINTOSH

DIMENSÕES

343 x 254 x 254 mm (vídeo/CPU /unidade de disco).

CPU

Motorola 68000, 7,83 MHz

MEMÓRIA

RAM de 128 K, ROM de 64 K

TELA

Monitor monocromático embutido, com 512 x 342 pixels, janelas.

INTERFACES

Mouse, impressora, unidade de disco externa, amplificador hi-fi, bus serial e analógico/digital.

LINGUAGENS DISPONÍVEIS

BASIC, COBOL, PASCAL

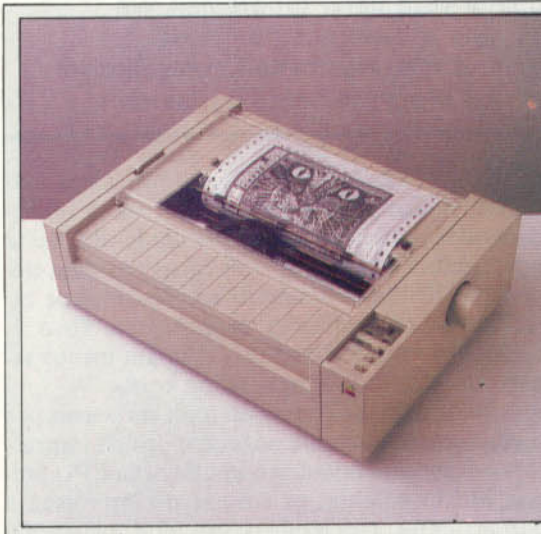
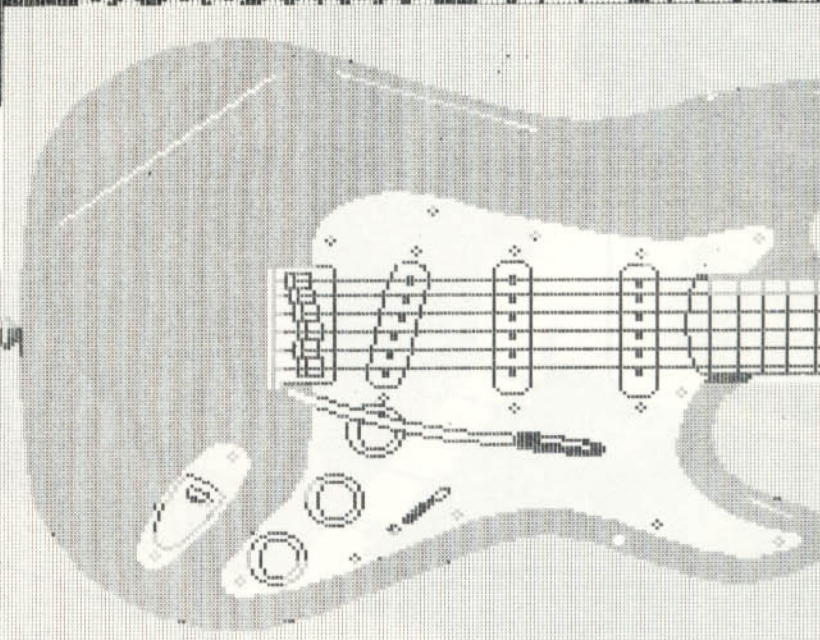
TECLADO

Tipo máquina de escrever, 59 teclas, teclado numérico opcional.

DOCUMENTAÇÃO

Existe um manual de operação com um audiocassete e uma instrução dirigida em disquete. Os manuais — MacPaint e MacWrite — são incluídos num só pacote, que combina disquete e cassete.

Melhores imagens

A *ImageWriter* é uma impressora gráfica serial capaz de imprimir até 120 caracteres por segundo. Sua velocidade é ainda mais evidente no modo gráfico: as figuras são traçadas por meio de um processo de "mapeamento a bit", seguindo o formato da tela.

IMPRESSORAS MATRICIAIS

Arte impressa
Estes grafismos mostram o efeito que se consegue com algumas impressoras matriciais. Cada agulha da cabeça de impressão é controlada individualmente, permitindo padrões complexos e satisfatórios. As imagens acima foram criadas pelo software PaintBox.

Quase todos os usuários de micros logo sentem falta de uma impressora. Usada para listar programas, ela permite a depuração de erros. Além disso, mostra-se imprescindível para o processamento de textos.

Se você pretende adquirir uma impressora matricial, alguns pontos a considerar são a velocidade de impressão e a qualidade do texto produzido. Os modelos mais caros possuem elementos adicionais, como diferentes tipos de caracteres e espaçamento proporcional (isto é, caracteres estreitos como o *i* ocupam menos espaço do que os maiores, como o *m*).

A velocidade de impressão (cps, caracteres por segundo) é importante porque o uso da impressora consome tempo do computador: a CPU precisa dividir-se entre as rotinas de impressão e outras tarefas. Um modelo caro, que opere a 200 cps, pode levar um minuto para imprimir a listagem de um programa extenso; modelos mais baratos, com velocidade de 30 cps, levariam mais de seis minutos para imprimir a mesma listagem. Em geral, durante todo esse tempo, o computador não pode ser usado para outra tarefa. Supera-se esse problema pelo uso de um buffer — uma memória auxiliar. Conectado entre o computador e a impressora, o buffer armazena os dados enquanto a impressora trabalha com eles, liberando o computador para outras operações.

As impressoras mais caras têm grandes buffers incorporados. Algumas armazenam só um caractere, ou só uma linha. A Alice, por exemplo, armazena 8 K de dados.

A velocidade de impressão apresentada pelo fabricante deve ser tomada com certa reserva. Como acontece com os números relativos ao consumo de combustível em automóveis, ela é dada para condições ideais, e com freqüência tem pouca semelhança com o desempenho real.

A velocidade é calculada para a impressão de uma única linha, composta de um mesmo caractere. Um texto normal, com seus caracteres variados, espaços, linhas e retornos do carro, diminui a velocidade final de impressão. Assim, quando estivesse imprimindo a listagem de um programa, a impressora com velocidade calculada em 160 cps faria uma média de 100 cps.

A qualidade dos caracteres produzidos no papel varia bastante de uma impressora para outra. Depende principalmente do número de agulhas da matriz de impressão — o mecanismo que forma os caracteres no papel, projetando algumas dessas agulhas.

A maioria dos modelos tem matriz de 9 x 7 agulhas. Já a Cannon PW 1080 utiliza matriz de 16 x 23 para produzir seus caracteres. Em conseqüência, os pontos não se distinguem e os caracteres têm aparência compacta e bem definida.

Para listagens de programas, a qualidade da impressão não é tão importante, mas para o processamento de texto e gráficos ela se revela fundamental.

A impressora matricial é um microcomputador dedicado à impressão. Utiliza chips de memória ROM e RAM e possui um microprocessador; pode, portanto, ser programada para executar outras tarefas além da impressão de textos, bastando para isso que se enviem códigos de controle especiais do micro para a impressora, ou que se instalem pequenas chaves, conhecidas como DIP (Dual In-line Packages), no interior da impressora. Os caracteres padrão ASCII, armazenados na memória da impressora, podem ser alterados para que se ajustem a alfabetos diferentes. No Brasil, produzem cedilha, til e acentos. Outros efeitos especiais incluem caracteres de largura dupla, texto em negrito (mais escuro e compacto) e diferentes espacejamentos de linha.

A Grafix MX 100 é uma das impressoras matriciais mais versáteis e tem inúmeros efeitos especiais de impressão. Imprime em caracteres itálicos e sublinha o texto automaticamente, além de permitir o espacejamento proporcional.

A linha de impressoras japonesas Epson tornou-se uma espécie de padrão industrial. Isso significa que a maioria dos softwares que necessitam de impressora — pacotes de processamento de texto, programas de faturamento etc. — implicam o uso de uma Epson. Em alguns casos, é necessário adaptar o software para uma impressora não compatível com a Epson.

Uma impressora barata pode ser ótima para a produção ocasional de listagens, mas é improvável que resista ao uso contínuo e diário num escritório.

O fator ruído raramente é levado em conta, mas, se você é do tipo que gosta de queimar as pestanas trabalhando à noite, saiba que algumas impressoras podem tornar-se irritantes no silêncio da madrugada.

As impressoras matriciais vêm equipadas com um mecanismo de tração que trabalha apenas com formulários contínuos — do tipo que tem furos dos lados. Para imprimir em folhas soltas é necessário um mecanismo de fricção semelhante ao da máquina de escrever comum.

A maioria das matriciais vem com interface paralela ou com interface serial RS232 C. Caso a impressora não possua a interface adequada a seu micro, procure saber junto ao revendedor se é possível acoplar a ela uma interface adicional. Mesmo com a interface adequada, é necessário o cabo correto para ligar a impressora ao computador.

Cedilha, til, acento
A Emília, da Elebra *(acima)* e a Grafix *(ao lado)* produzem cedilha, til e acentos, sinais imprecindíveis para a escrita em português. Além disso, conseguem muitos efeitos especiais. (*Abaixo,* teste de impressão da Grafix.)

MICROCOMPUTADOR	MICROCOMPUTADOR	MICROCOMPUTADOR
MICROCOMPUTADOR	MICROCOMPUTADOR	MICROCOMPUTADOR
MICROCOMPUTADOR	MICROCOMPUTADOR	MICROCOMPUTADOR
MICROCOMPUTADOR	*MICROCOMPUTADOR*	MICROCOMPUTADOR
MICROCOMPUTADOR	MICROCOMPUTADOR	MICROCOMPUTADOR
MICROCOMPUTADOR	MICROCOMPUTADOR	**MICROCOMPUTADOR**
MICROCOMPUTADOR	**MICROCOMPUTADOR**	

ARQUIVOS SEQÜENCIAIS

Este capítulo é dedicado à discussão dos métodos de utilização dos arquivos seqüenciais em seus programas e mostra como superar algumas das limitações próprias do sistema.

O arquivo seqüencial consiste num grupo de registros gravados em bloco no disquete ou na fita cassete. Esse tipo de armazenamento apresenta limitações no que diz respeito a acesso (leitura) e atualização de dados.

O acesso a qualquer registro exige a leitura de todos os dados precedentes. E a atualização, em geral, obriga o usuário a munir-se de uma cópia do arquivo até o ponto em que as mudanças são necessárias; feito isso, ele pode acrescentar as alterações e prosseguir copiando o arquivo original.

A organização das informações no arquivo depende do programador e da aplicação que terá. Um arquivo que contenha apenas textos resume-se a uma seqüência de códigos ASCII, seguida por um marcador de fim de arquivo. No entanto, se o arquivo contiver um banco de dados, como, por exemplo, um catálogo de livros, as informações devem ser organizadas de outra forma. A mais comum é dividir o arquivo em *registros* e *campos*. Cada livro tem seu registro no arquivo e dentro de cada registro há vários campos, tais como o título do livro, o autor, a editora e assim por diante. Num arquivo seqüencial, essas divisões devem ser marcadas por meio de caracteres especiais colocados entre os itens de dados.

Para tanto, costuma-se usar o caractere "retorno do carro" (código ASCII número 13), que atua como marcador entre os campos e os registros. Uma vez que o arquivo terá o mesmo número de campos para cada registro, é fácil para o programa localizar onde termina um registro e começa outro. Depois de criado, o arquivo seqüencial deve dar condições de acesso e atualização. As operações básicas na manipulação de arquivos são: acessar (ler), acrescentar, eliminar e modificar (editar) registros. Os diagramas mostram as várias maneiras de realizar tais operações com os arquivos seqüenciais: à medida que faz a leitura (obrigatoriamente na ordem), o programa cria nova cópia do arquivo. Nessa cópia gravam-se as alterações, introduzidas quando chega sua vez. Por fim, a cópia substitui o arquivo antigo, que é descartado, ou mantido como cópia de reserva (back-up).

Essas técnicas simples são a base de todas as rotinas de arquivamento seqüencial. Partem do princípio de que o sistema operacional pode ter dois arquivos diferentes abertos ao mesmo tempo — para ler de um e gravar no outro simultaneamente. Isso, no entanto, só é possível em alguns sistemas a disco e nos micros baseados em fitas cassete com dois gravadores conectados. Alguns modelos de micro dispõem de interfaces para dois gravadores cassete. As máquinas com saída para um único cassete só comportam arquivos pequenos — o suficiente para que sejam lidos na íntegra na memória e aí processados.

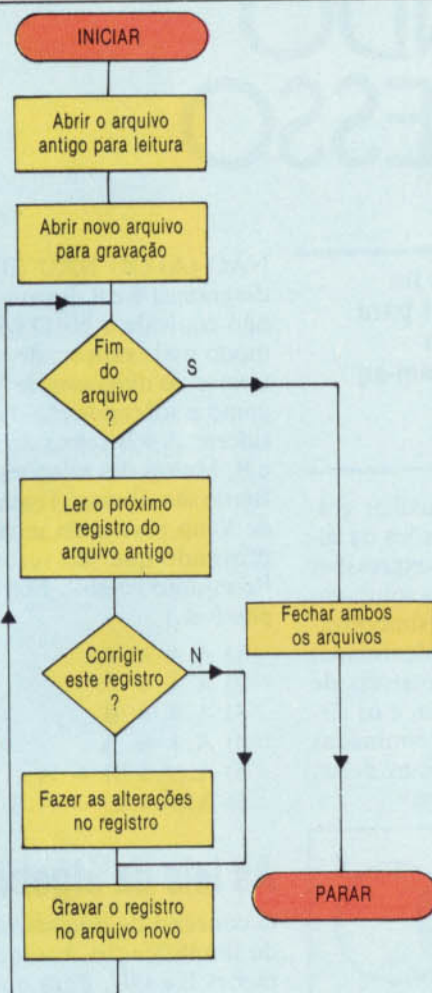
Esses métodos de organização dos arquivos são vantajosos também porque oferecem duas cópias do arquivo: uma do antigo, como era antes da atualização, e outra do novo, com as alterações. A guarda de ambos é precaução necessária porque, se algo acontecer ao arquivo, pode-se recuperá-lo pela simples atualização (de uma geração) da cópia. Na maioria das empresas costuma-se guardar três gerações de cada arquivo, que no jargão técnico recebem os nomes de filho, pai e avô.

Essas técnicas são ideais para arquivos grandes, que não cabem na memória do computador — em geral, acessa-se, a cada vez, apenas uma parte do arquivo (alguns registros). No caso de arquivos pequenos, obtém-se melhor desempenho lendo todo o arquivo e colocando-o numa área da memória para aí ser processado. Todas as operações com arquivo se desenvolvem em alta velocidade na memória; em seguida, o novo arquivo é gravado na íntegra no disquete ou na fita cassete.

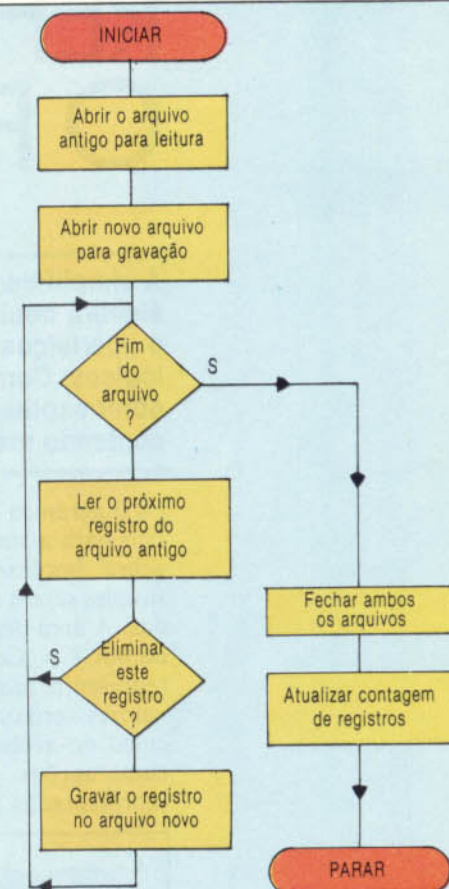
Com esse método corre-se um grande risco — as mudanças nos arquivos se tornam permanentes apenas quando as informações são gravadas de volta na fita ou no disquete; assim, podem-se perder os dados se o programa "travar" ou se o computador sofrer algum acidente ou for desligado durante a operação. Se você estiver usando programas que funcionem dessa maneira, certifique-se de que vêm sendo feitas cópias do arquivo para armazená-lo e verifique se foi produzida a cópia atualizada antes de terminar o programa.

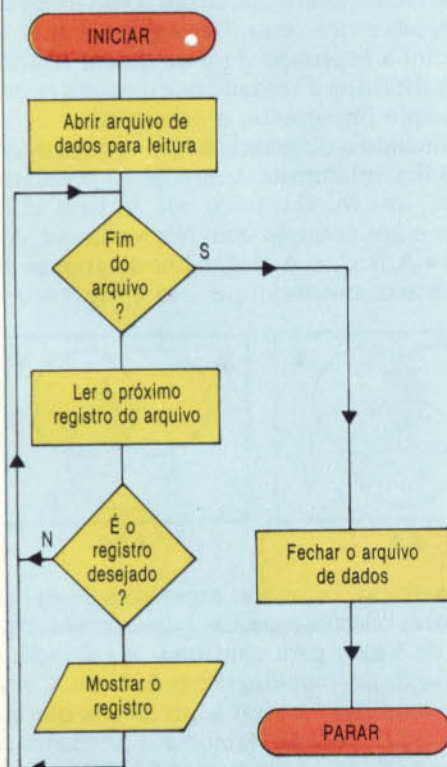
Um pouco de experiência em lidar com arquivos seqüenciais vai mostrar-lhe que as técnicas envolvidas, apesar de trabalhosas, são úteis. Em muitos sistemas pequenos, o arquivo seqüencial é a única estrutura de arquivos disponível. Outras técnicas que complementam os arquivos seqüenciais, com acesso e atualização simples e rápidos, encontram-se no estudo dos arquivos de acesso aleatório.

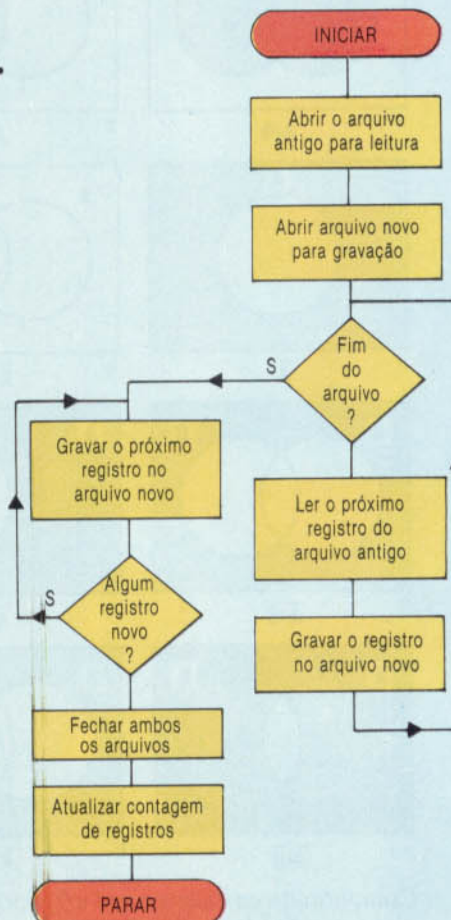
Correção de registros

Para corrigir um registro (alterar as informações nele contidas), começa-se copiando todos os registros precedentes em novo arquivo. Quando se chegar ao registro a ser alterado, o programa faz a atualização. Depois de atualizado, o registro é gravado no arquivo novo e, em seguida, gravam-se todos os registros subseqüentes do arquivo antigo. Qualquer número de registros pode ser atualizado numa só operação de busca por meio do arquivo.

Eliminação de registros

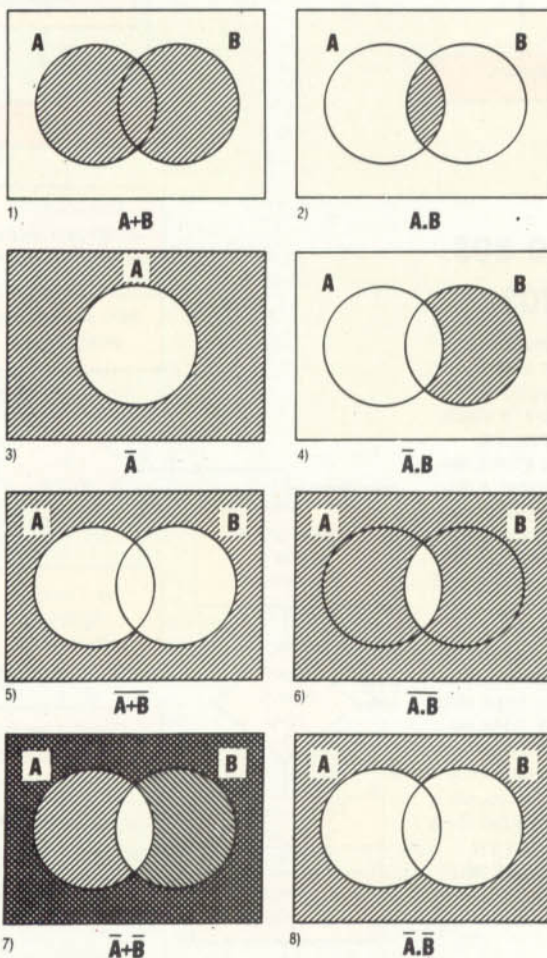
Para eliminar um registro, o programa faz a leitura e a cópia do arquivo, até chegar ao registro desejado. Este é lido, mas não copiado no arquivo novo. Em seguida, os registros remanescentes são lidos e copiados no novo arquivo. Vários registros podem ser eliminados numa única pesquisa. Da mesma forma que na inserção de registros, aqui também é essencial que se atualize sua contagem imediatamente.

INICIAR
Abrir arquivo de dados para leitura
Fim do arquivo ?
S
Ler o próximo registro do arquivo
É o registro desejado ?
N
Mostrar o registro
Fechar o arquivo de dados
PARAR

Acesso aos registros

Os arquivos seqüenciais são os menos indicados para as aplicações que exigem a leitura de alguns registros apenas. Isso porque todo o arquivo tem de ser lido de novo, o que consome muito tempo. Se você estiver procurando determinado registro numa lista de nomes, o programa percorrerá um loop, examinando cada registro e prosseguindo no loop sempre que não preencher a condição. Se forem vários registros, eles podem ser lidos um após o outro, mas sempre na ordem em que aparecem no arquivo. Daí a importância de organizá-los em alguma ordem (por exemplo, alfabética) antes de armazená-los.

Inserção de registros

Há duas maneiras de inserir registros num arquivo. Algumas versões do BASIC têm um comando APPEND (anexar) que permite acrescentar os registros diretamente no fim do arquivo. Na ausência desse comando é necessário ler todo o arquivo e produzir uma cópia em arquivo novo. Em vez de fechar o arquivo novo, escrevem-se os registros no final e fecham-se os dois arquivos. Em ambos os casos, é preciso atualizar a contagem dos registros. Se a contagem for armazenada no arquivo, a rotina de atualização deve certificar-se de que o novo valor está gravado no registro, a fim de evitar a perda de informações.

REFINANDO O PROCESSO

A simplificação das expressões na álgebra booleana é fundamental para o aperfeiçoamento dos circuitos lógicos. Com esse recurso, podem-se obter expressões equivalentes contendo menos operadores.

Os diagramas de Venn são um útil auxiliar gráfico para a simplificação das expressões da álgebra booleana, permitindo que expressões simples sejam desenhadas como áreas sombreadas. A área dentro de um retângulo (simbolizada por 1 = Conjunto Universo ou Identidade) representa todas as combinações possíveis de valores-verdade dos dados de entrada, e os círculos no retângulo representam determinadas combinações. Aqui mostramos algumas delas, representadas por diagramas de Venn:

Comparando os diagramas 5 e 7, nota-se de imediato que NÃO (A OU B) não é o mesmo que NÃO (A) OU NÃO (B). Do mesmo modo, os diagramas 6 e 8 demonstram que NÃO (A E B) não equivale a NÃO (A) E NÃO (B). Talvez o modo mais simples de imaginar o E e o OU em termos do diagrama de Venn seja pensar em A.B como a sobreposição da área A à área B, e considerar A + B como as áreas combinadas de A e B. Muitas das relações existentes na álgebra de Boole são óbvias. Tente construir um diagrama de Venn para cada uma das relações seguintes, provando que são verdadeiras. (0 representa o "conjunto vazio", isto é, uma combinação impossível.)

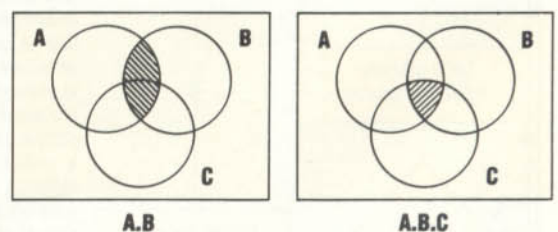
1) $A.A = A$
2) $A.\overline{A} = 0$
3) $A.0 = 0$
4) $A.1 = A$
5) $A.(A+B) = A$
6) $A.(\overline{A}+B) = A.B$

As leis da álgebra booleana

O conceito de *dualidade* é um eficiente recurso de simplificação, baseado na simetria das operações E e OU. Para obter a expressão dual de qualquer relação verdadeira, transforme todos os Es em OUs e vice-versa, assim como todos os zeros em uns e vice-versa. Exemplo: $A + A.B = A$ constitui a expressão dual da quinta relação de nossa lista. Ela é verdadeira e demonstra outro princípio importante, a *absorção*.

Examinando o diagrama, é fácil ver que o termo A.B fica totalmente dentro de A; constata-se, assim, que foi absorvido por A. Essa idéia aplica-se a um exemplo com três variáveis, como $A.B + A.B.C = A.B$. Os dois diagramas de Venn, abaixo, mostram que isso é verdadeiro.

A.B

A.B.C

Como exercício, escreva as expressões duais das cinco outras relações especiais e desenhe seus diagramas de Venn, para confirmar a validade.

Observe de novo os diagramas de Venn. Comparando o quinto e o oitavo, percebe-se que esta importante relação também é verdadeira: $\overline{A+B} = \overline{A}.\overline{B}$. Já os diagramas 6 e 7 mostram que $\overline{A.B} = \overline{A}+\overline{B}$. Essas duas relações consti-

ituem as *leis de De Morgan* e incidem também sobre expressões mais complexas, como as de três variáveis ($\overline{A+B+C} = \overline{A}.\overline{B}.\overline{C}$ e $\overline{A.B.C} = \overline{A}+\overline{B}+\overline{C}$). As leis de De Morgan podem ser aplicadas em estágios:

$(\overline{A+B}).\overline{C} =$
$= \overline{A}.\overline{B}.\overline{C}$ (usando-se a lei de De Morgan nos parênteses); ou =
$= \overline{A+B+C}$ (recombinando com o uso da lei de De Morgan).

Três outras regras aplicam-se, como na álgebra normal, à álgebra booleana. A *lei associativa* permite o deslocamento de parênteses:

$(A.B).C = A.(B.C) = A.B.C$
$(A+B)+C = A+(B+C) = A+B+C$

A ordem na qual as letras são escritas pode ser mudada, de acordo com a *lei comutativa:*

$A.B = B.A$
$A+B = B+A$

A *lei distributiva* permite a distribuição do multiplicador pelos parênteses:

$A.(B+C) = A.B+A.C$

Exemplos de simplificação

1) $(\overline{A+B}+\overline{A}.B).B =$
$= (\overline{A}.\overline{B}+\overline{A}.B).B$ (de De Morgan);
$= \overline{A}.\overline{B}.B+\overline{A}.B.B$ (lei distributiva);
$= 0+\overline{A}.B$ ($\overline{B}.B = 0$, $B.B = B$);
$= \overline{A}.B$;

2) $\overline{A}.\overline{B}+\overline{A}.B+A.B =$
$= \overline{A}.(\overline{B}+B)+A.B$ (lei distributiva);
$= \overline{A}+A.B$ ($\overline{B}+B = 1$);
$= \overline{A}+B$ (dual da relação 6);

3) $\overline{\overline{A}+B}+\overline{\overline{A}+\overline{B}}+\overline{A}.B$
$= \overline{\overline{A}}.\overline{B}+\overline{\overline{A}}.\overline{\overline{B}}+\overline{A}.B$ (de De Morgan);
$= A.\overline{B}+A.B+\overline{A}.B$ ($\overline{\overline{A}} = A$);
$= A.(\overline{B}+B)+\overline{A}.B$ (lei distributiva);
$= A+\overline{A}.B$ ($\overline{B}+B = 1$);
$= A+B$ (dual da relação 6).

Porta OU exclusivo simplificada

Na matéria anterior desta seção examinamos um circuito não simplificado para porta OU exclusivo. Voltamos agora a abordar a questão, desta vez com o recurso de simplificar a expressão booleana e, com isso, também o circuito. A tabela de validação para a porta OU exclusivo é:

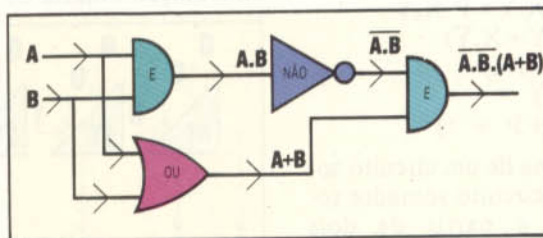
ENTRADA		SAÍDA
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

A partir da tabela de validação, determinamos que $C = \overline{A}.B+A.\overline{B}$. É pouca a simplificação que se pode fazer e seria necessário um circuito de cinco portas para executar a expressão. Contudo, há um modo alternativo de abordar o problema. Segundo a tabela de validação, C é 1 se A e B não forem ambos 1 ou ambos 0. Em termos booleanos, pode-se escrever uma expressão alternativa para C:

$$C = \overline{A.B+\overline{A}.\overline{B}}$$

Usando as leis de De Morgan sucessivamente, simplifica-se o circuito e obtém-se:

$$C = (\overline{A.B}).(\overline{\overline{A}.\overline{B}}) = \overline{A.B}.(A+B)$$

Essa expressão exige apenas quatro portas:

Circuito somador total

Já examinamos o processo de adição binária e projetamos um circuito simples (o somador parcial) para somar dois bits e produzir duas saídas para a soma e os dígitos de transporte da resposta. Se denominarmos X a primeira entrada e Y a segunda, podemos verificar pela tabela de validação para um somador parcial que a saída (S) da soma — ou seja, a resposta — pode ser representada pela expressão: $S = \overline{X}.Y+X.\overline{Y}$. Usando a lei de De Morgan, essa expressão pode ser simplificada para $S = \overline{X.Y}.(X+Y)$. A saída (C) de transporte é $C = X.Y$.

Na aritmética binária, três dígitos precisam ser acrescentados em qualquer das colunas do resultado da adição. Além dos dois dígitos, há um transporte das colunas anteriores a ser incluído. Para reproduzir o processo de adição binária, devemos desenhar um circuito com três entradas e duas saídas. Se chamarmos P o transporte da coluna anterior, a tabela de validação para o somador total (ST) será:

ENTRADAS			SAÍDAS	
P	X	Y	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Tomando os exemplos em que S = 1, pode-se formar uma expressão para S a partir da tabela de validação:

$$S = \overline{P}.\overline{X}.Y + \overline{P}.X.\overline{Y} + P.\overline{X}.\overline{Y} + P.X.Y$$

Utilizando as regras vistas acima, podemos simplificar esta expressão:

$$S = \overline{P}.(\overline{X}.Y + X.\overline{Y}) + P.(\overline{X}.\overline{Y} + X.Y)$$

(lei distributiva);

$$S = \overline{P}.(\overline{X}.Y + X.\overline{Y}) + P.(\overline{\overline{X}.Y.X.\overline{Y}})$$

(de De Morgan)

De modo análogo, podemos formar uma expressão para C a partir da tabela de validação:

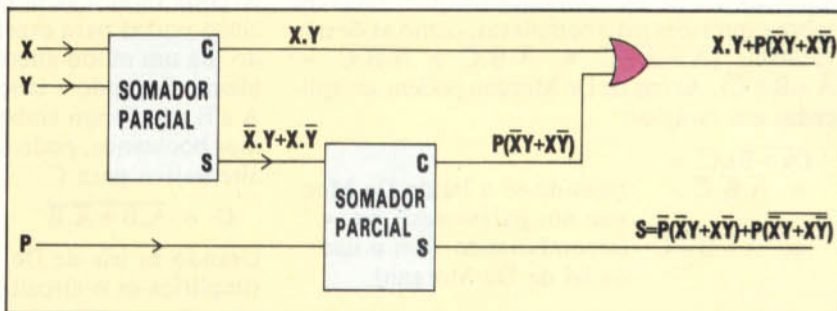
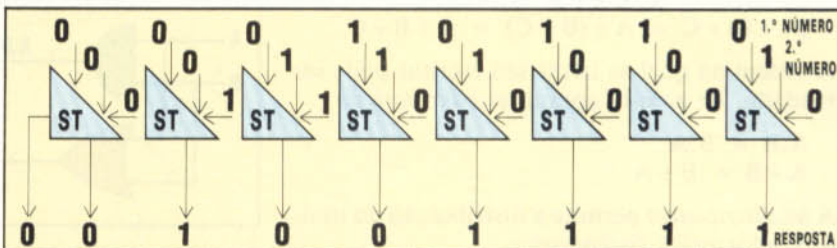
$$C = \overline{P}.X.Y + P.\overline{X}.Y + P.X.\overline{Y} + P.X.Y$$

$$C = X.Y.(\overline{P} + P) + P.(\overline{X}.Y + X.\overline{Y})$$

(lei distributiva)

$$C = X.Y + P.(\overline{X}.Y + X.\overline{Y})$$

$(\overline{P} + P = 1)$

$\overline{X}.Y + X.\overline{Y}$ é a saída da soma de um circuito somador parcial. Assim, um circuito somador total pode desenvolver-se a partir de dois somadores parciais.

Uma série de oito somadores totais se combina numa unidade lógica aritmética para executar uma adição binária de dois números de 8 bits:

Exercício 3

1) Simplifique estas expressões:

a) $A.(\overline{A} + \overline{B})$

b) $X + Y.(X + Y) + X.(\overline{X} + Y)$

c) $P.Q + \overline{P}.Q + \overline{P}.\overline{Q}$

d) $\overline{X + \overline{Y}.\overline{Z}} + \overline{Z}.Y$

2) O alarma de um carro possui interruptores para ligar/desligar nas duas portas da frente e um interruptor geral. O alarma soará se qualquer uma ou ambas as portas forem abertas quando o interruptor geral estiver ligado. Faça uma tabela de validação mostrando as três entradas (porta A, porta B e o interruptor geral), e a saída de alarma. Use a tabela de validação para escrever uma expressão booleana para o alarma soando. Desenhe um circuito lógico para o sistema de alarma.

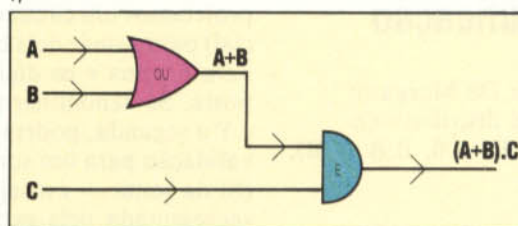
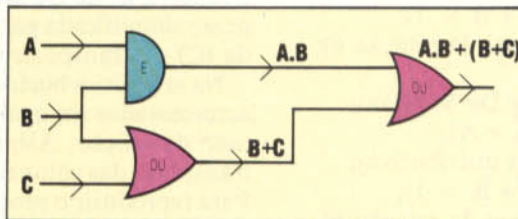
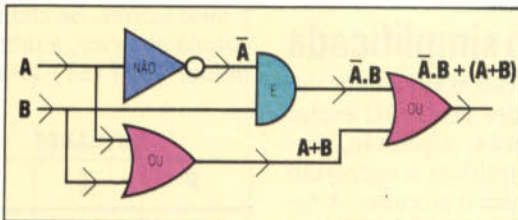
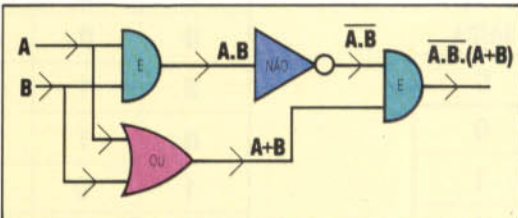
3) A lâmpada de uma saleta é operada por meio de um interruptor na porta, um interruptor na base da escada e outro no topo da escada. Desenhe um circuito lógico adequado.

4) Você está preso numa ilha deserta com duas outras pessoas. Uma delas sempre fala a verdade, a outra sempre mente. Para sua própria sobrevivência, é importante descobrir qual delas fala a verdade. Há uma série de perguntas que você pode fazer a cada uma delas para determinar sua identidade. Desenhe as tabelas de validação para investigar as respostas possíveis. Eis um exemplo para você começar:

"Você sempre diz a verdade?"

		RESPOSTAS POSSÍVEIS	
		SIM	NÃO
POSSÍVEL IDENTIDADE DE QUEM RESPONDE	MENTIROSO	1	0
	SINCERO	1	0

Respostas do exercício 2

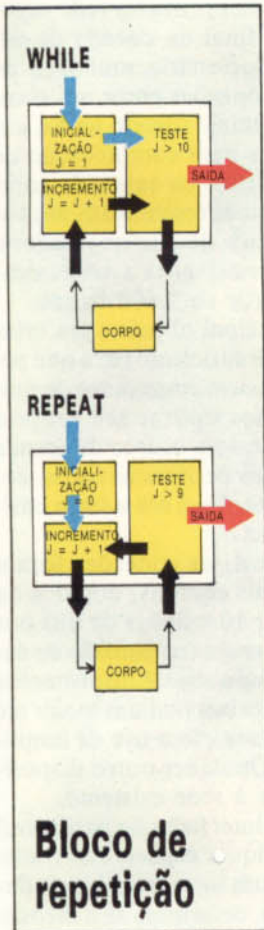
1)

a)

b)

c)

d)

2a) $C = A.B$

b) $S = \overline{A.B}(A + B)$

3a) $X = (A + B).(B.C + C)$

b) $X = \overline{A.\overline{B}} + \overline{B}$

LINHAS DE LOOP

Bloco de repetição

Os fluxogramas convencionais nem sempre representam com clareza as estruturas de loop. Esta matéria estuda o funcionamento dos loops e apresenta novo símbolo para o fluxograma — os blocos de repetição.

A repetição, ou loop, é uma das estruturas essenciais em qualquer linguagem de programação. Usa-se o loop quando uma decisão desvia o fluxo do programa para um trajeto que retorna à decisão inicial. O loop é, assim, uma estrutura responsável pela execução repetida de um corpo de instruções.

Abrangendo cerca de 50% de toda a atividade de processamento de dados, os loops merecem um estudo detalhado, sobretudo no que se refere a seu efeito sobre a estrutura geral do programa-algoritmo e às várias formas de sua construção e classificação.

Os loops costumam ser divididos em dois tipos, dependendo de sua semelhança com as duas estruturas de loop existentes nas linguagens de mais alto nível: REPEAT — UNTIL (repita-até) e WHILE — ENDWHILE (enquanto-fim).

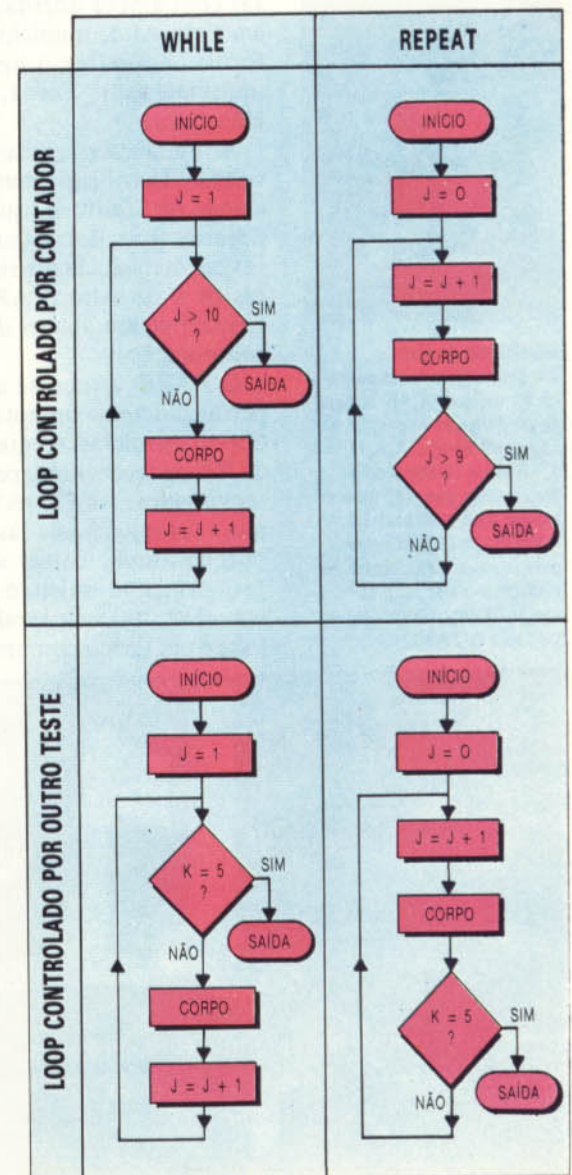
Ambos os tipos são usados em PASCAL e o loop REPEAT existe em algumas versões do BASIC. No BASIC comum, o loop é feito da forma: IF-THEN-GOTO, que implica montar uma rotina de teste com quatro instruções no mínimo. Nas funções REPEAT e WHILE, esta rotina já está incorporada. Os dois tipos diferem quanto ao posicionamento dos testes de saída: no REPEAT, o teste vem no final do corpo do loop, enquanto no WHILE se encontra no início. Assim, o corpo de um loop REPEAT será sempre executado pelo menos uma vez; não ocorre o mesmo com o loop WHILE. O fluxograma de blocos permite visualizar essa diferença.

Outra forma de classificar os loops se refere ao controle de saída: pode ser realizado pelo próprio contador do loop ou por outra condição qualquer. Essa diferença evidencia-se menos num fluxograma de tipo linear convencional. Em fluxogramas desse tipo, a descrição dos loops é pouco útil: assemelha-se a uma simples ramificação e com freqüência exige o exame detalhado do algoritmo para distinguir um do outro.

Existe uma notação mais clara, chamada "bloco de repetição", que marca o início de um loop e elimina a confusão entre loop e ramificação. Essa notação consiste em três blocos interligados: o primeiro mostra o início da contagem; o segundo, o incremento da contagem; e o terceiro contém o teste de saída do loop.

Tanto os loops REPEAT como os WHILE podem ser escritos dessa forma, mas diferem quanto ao fluxo do programa pelos blocos; o bloco de teste indica se o loop é controlado pelo contador ou não. O diagrama evidencia isso.

Num loop do tipo REPEAT, o fluxo das instruções é: início-corpo-teste-corpo-teste. Um loop do tipo WHILE segue o esquema início-teste-corpo-teste-corpo. As linhas que representam o fluxo saem e retornam ao diagrama de repetição, com o corpo do loop "suspenso" por tais linhas de controle.

Classificação dos loops

INVESTINDO NO FUTURO

Quando a Xerox — a maior vendedora mundial de copiadoras — resolveu lançar-se no campo da automação de escritórios, sua reputação era tal que as pessoas já usavam "xerocar" como sinônimo de "fotocopiar".

No início dos anos 70, a Xerox planejou um intenso programa de pesquisas para que um sonho se tornasse realidade: dispor de informações em quantidade tão abundante no escritório que seu fluxo semelhasse uma corrente elétrica. Ela reuniu nos Estados Unidos uma equipe de pesquisas com ampla liberdade de ação, trabalhando em Palo Alto, município de Santa Clara, Califórnia, no extremo oposto das instalações da multinacional Xerox, em Rochester, New Hampshire.

A mudança para Santa Clara foi muito proveitosa. Localizado perto do campus da Universidade de Stanford, que possui um importante departamento de computação dedicado a pesquisas de inteligência artificial, o Centro de Pesquisas de Palo Alto (PARC, Palo Alto Research Center) atraiu alguns dos maiores cérebros em computação.

Nascimento do Lisa
Um dos maiores feitos da PARC foi a criação do STAR, sistema de programação que utiliza a linguagem SMALLTALK. O STAR opera combinando programas e dados no mesmo arquivo para processamento. A tecnologia Lisa da Apple deve muito a esse sistema. Na verdade, a maior parte da equipe de desenvolvimento do Lisa veio do PARC.

O PARC tornou-se um centro de cultura especializado em computação, criando um jargão que só os iniciados entendem. Muitos produtos da Xerox receberam apelidos enquanto eram desenvolvidos. As 820 séries de micros, por exemplo, foram chamadas de WORM (verme), porque iriam "comer a Apple" (maçã).

O principal objetivo da nova equipe era desenvolver uma rede local (LAN, Local Area Net). Esse é um termo comum hoje em dia, mas quando a Xerox estabeleceu sua primeira rede experimental no Havaí, no final da década de 60, representava uma revolucionária mudança de conceito. Para que as conexões entre um computador central e um terminal fossem feitas, era necessário instalar cabos para comunicação de alta velocidade. Além disso, os cabos de comprimento superior a 20 m apresentavam muitos problemas. A rede pública de telefone poderia ser utilizada, mas isso restringiria a comunicação de dados à velocidade de 9.600 bauds.

Em Palo Alto, o principal objetivo era criar uma rede com velocidade suficiente para que pequenos computadores fossem conectados de modo que o usuário pudesse utilizar seu próprio aparelho, além de ter acesso a outros de grande porte, a seus discos e mais periféricos caros, como plotters e impressoras. Essa era a idéia básica da rede local Ethernet.

No sistema da Ethernet, as conexões foram feitas com cabos co-axiais comuns, dotados da capacidade de transmitir 10 milhões de bits por segundo. Esse sistema permite transmissão de dados digitalizados, incluindo sons e informações gráficas. Além disso, os cabos podiam medir até 500 m sem que fosse necessário o uso de amplificadores de repetição. Qualquer outro dispositivo poderia se conectar à rede existente.

A rede física (cabos e interfaces de hardware) é passiva: dados de qualquer espécie são transportados através dela e um interpretador de dados, em cada dispositivo, determina se a mensagem se destina àquele dispositivo. Se for assim, o interpretador decodifica a mensagem, apresentando-a de tal forma que aquele dispositivo — seja ele um microcomputador, uma impressora, um plotter ou qualquer outro — possa utilizá-la.

Por volta da metade dos anos 70, a Ethernet já estava funcionando. A Xerox percebeu que, se obtivesse ajuda de outras empresas, poderia fazer desse sistema um modelo universal de comunicação entre computadores. Ela levou seus projetos para a IBM, que não quis participar. A Digital Equipment Corporation, contudo, aderiu.

Em 1975, a Xerox também conseguiu a cooperação da Intel, fabricante de chips, que passou a produzir um chip dedicado à interpretação de dados. A rede Ethernet foi testada num escritório experimental e num complexo empresarial na Suécia. Depois dos testes, considerados satisfatórios, outros fabricantes adotaram o sistema. Tornou-se um modelo internacional admitido por empresas como a Hewlett-Packard e a ICL, na Grã-Bretanha, a Siemens, na Alemanha, e a Olivetti, na Itália.

OS PESOS LEVES

Em 1981, Adam Osborne lançou o primeiro micro portátil. Quatro anos depois, o Osborne e os outros modelos nele inspirados parecem incômodos e pesados, diante de equipamentos menores que uma lista telefônica.

O que é um microcomputador "portátil"? Não é tão simples responder: o alcance do qualificativo teve de ser revisto desde o aparecimento da última geração de computadores "de mão". Na verdade, os micros portáteis introduzidos no início dos anos 80 podem agora ser chamados, no máximo, de micros "transportáveis". Em 1985, os microcomputadores portáteis são os que possuem sua própria fonte de energia, tela e dispositivos para armazenamento de dados num conjunto menor que uma lista telefônica.

O Epson HX-20 foi o primeiro a oferecer esse grau de "portabilidade". Hoje, porém, seu pequeno visor de cristal líquido com capacidade de vinte caracteres por quatro linhas denuncia a idade do projeto. Os portáteis mais recentes, como o Tandy 100, o NEC PC 8201-A e o Olivetti M 10, têm preços similares, mas podem mostrar quatro vezes mais caracteres no visor.

Então, o que podem fazer esses computadores? Quais as suas vantagens e desvantagens em relação aos micros convencionais de mesa?

O motivo mais óbvio para se adquirir um micro portátil é o acesso total à capacidade de processamento a qualquer hora e em qualquer lugar. Muita gente passa grande parte do tempo longe de seu computador; horas improdutivas são gastas em escritórios de clientes, quartos de hotel, aeroportos e trens. O computador portátil permite que todo esse tempo seja aproveitado.

A última geração de portáteis oferece recursos de computação suficientes para trabalhos científicos ou de engenharia, cálculos, gerenciamento financeiro e processamento de texto — praticamente para todas as aplicações em que se usam computadores pessoais convencionais. Além disso, podem ser conectados a impressoras, modems e outros periféricos.

Os computadores portáteis trazem normalmente pelo menos três programas incorporados em chips de memória — um interpretador BASIC, um processador de texto e um software de telecomunicações. O Tandy 100 e o Olivetti possuem, além destes, programas de endereço e de agenda, que permitem encontrar endereços, números de telefone e compromissos diários.

O programa de telecomunicações é extremamente importante, pois permite que o portátil tenha acesso a outros micros e bancos de dados remotos pela rede telefônica. Com esse recurso, o micro pode transformar-se num terminal de telex ou num transmissor-receptor de correio eletrônico: basta que se utilize um modem ou um acoplador acústico. Dessa maneira, um executivo em viagem pode manter-se em contato com o escritório central, ou um jornalista em trânsito pode escrever seus artigos no computador e transmiti-los imediatamente ao computador do jornal.

Os portáteis mais caros, como o Sharp PC-5000 e o Epson PX-8, utilizam os sistemas operacionais MS-DOS e CP/M, adotados por seus equivalentes de mesa. Estão, portanto, capacitados a rodar inúmeros programas administrativos e financeiros.

Em viagem
O uso de computadores durante as viagens vem se tornando um hábito cada vez mais comum entre profissionais. A nova geração de micros portáteis permite que os homens de negócios ganhem um tempo extra na confecção de textos, enquanto, por exemplo, correm de táxi para o aeroporto. Já os vendedores podem calcular na hora um orçamento — operação que normalmente exige vários dias para ser realizada.

Os executivos em trânsito podem, usando um modem e a rede telefônica, transmitir dados à sede de sua empresa ou, retornando ao escritório no fim do dia, enviar os dados diretamente a um computador de maior porte.

O Epson PX-8 traz o processador de texto WordStar já residente em seus chips de ROM. O Sharp utiliza cartuchos de memória de bolha, que garantem, cada um, 128 Kbytes de memória extra. Esses cartuchos lidam com os dados mais rapidamente que as unidades de disco.

Os programas aplicativos dos portáteis mais baratos são normalmente carregados em sua RAM a partir de fitas cassete, um processo muito mais lento do que o dos cartuchos de memória de bolha ou dos disquetes. O NEC PC 8201-A vem com uma fita cassete que contém aplicativos para cálculos, formatação de textos, gerenciamento de investimentos e avaliações de empréstimos. O programa de cálculos transforma a máquina numa calculadora capaz de memorizar até 99 entradas. O formatador prepara para impressão os materiais elaborados pelo editor de texto, especificando a largura das margens, dividindo o texto em páginas, numerando as páginas etc. Com o programa de investimentos, o micro pode analisar um conjunto de até cinqüenta deles, calculando ganhos e perdas.

O micro portátil tem ainda como características alimentação por baterias, visor próprio e o processador de texto e software de comunicações que traz na ROM.

Isso não acontece com o Apple II*c* e o Apricot, geralmente anunciados como portáteis. Esses micros precisam ser ligados à rede elétrica, conectados a um monitor de vídeo e seus programas são carregados na RAM a partir de um disco. Apesar do tamanho e do peso reduzidos, estão mais próximos dos computadores de mesa do que dos verdadeiros portáteis, pertencendo à categoria "transportável".

Além de suas baterias principais, os micros portáteis possuem pequenas pilhas de cádmio e níquel, que constituem uma fonte de energia em caso de emergência. Isso é essencial, uma vez que todos os dados seriam perdidos no caso de acabarem ou falharem as baterias.

A maioria dos portáteis tem também uma interface para leitura do código de barras, permitindo sua utilização no controle de estoques. Ao passar-se a leitora sobre o código de barras impresso na embalagem dos produtos, ela decodifica as informações de preços e datas. Tais informações são processadas pelo computador, facilitando aos lojistas a manutenção de um arquivo atualizado das mercadorias.

Tanto o Tandy Modelo 100 como o NEC PC 8201-A e o Olivetti M 10, mostrados nestas páginas, dispõem de leitoras de código de barra, o que não chega a surpreender: produzidos pela mesma fábrica japonesa, os três apresentam muitas semelhanças. Existem, porém, diferenças significativas entre eles. O Olivetti é o único com tela inclinável. O NEC tem menos software incorporado, mas sua memória pode ser expandi-

Epson HX-20
Embora dotado de um visor pequeno, o HX-20 tem a vantagem de trazer, incorporados, um gravador cassete e uma miniimpressora. Um processador de texto com recursos básicos também está incluído.

Casio FP-200
O Casio é o mais barato de todos os computadores portáteis. Todavia, não possui processador de texto, fornecendo, em vez disso, uma planilha eletrônica rudimentar.

da até 64 Kbytes: o dobro da possibilidade de expansão de memória do Tandy e do Olivetti. Além disso, o NEC pode utilizar cartuchos sobressalentes com 32 Kbytes de memória, que retêm os dados mesmo quando removidos do computador.

Assim como as máquinas de escrever portáteis não tornaram obsoletas as pesadas máquinas de escritório, os micros portáteis não pretendem tomar o lugar dos microcomputadores de mesa. Para começar, seus pequenos visores de cristal líquido (LCD) os tornam pouco adequados a longas sessões de trabalho. O LCD é de leitura mais difícil e responde mais lentamente à digitação que os vídeos de raios catódicos.

Outra desvantagem dos computadores portáteis são os teclados planos, mais cansativos de usar. E os modelos de preço mais acessível não rodam os programas aplicativos comerciais populares.

Apesar de tudo isso, é inegável que os computadores "pesos leves" vieram para ficar. Com o uso generalizado dos micros, um número cada vez maior de pessoas está descobrindo que, enquanto um computador pode ajudar a administrar suas atividades com maior eficiência, os portáteis lhes permitem ter acesso a esse recurso estejam onde estiverem. Não vai demorar muito para que os micros portáteis se tornem tão populares quanto as calculadoras de bolso.

Epson PX-8
Esta máquina pode rodar softwares comerciais com o sistema CP/M, inclusive o processador de texto WordStar, que vem junto com a máquina.

Tandy TRS-80 100/NEC PC 8201-A/Olivetti M 10
Três versões diferentes do mesmo micro. Possuem em comum um excelente processador de texto incorporado e razoável gama de interfaces. Também na ilustração, um modem a bateria da Olivetti.

Modelo	Memória padrão	Memória máxima	Visor	Peso
Casio FP-200	8 K	32 K	8x20	1,400 kg
Epson HX-20	16 K	32 K	4x20	1,800 kg
Epson PX-8	64 K	64 K + 120 K*	8x80	2,300 kg
NEC PC 8201-A	16 K	64 K + 32 K*	8x40	1,800 kg
Olivetti M 10	8 K	32 K	8x40	1,800 kg
Tandy TRS-80 Modelo 100	8 K	32 K	8x40	1,800 kg

* O NEC aceita um cartucho de RAM com 32 K e o Epson PX-8, um disco com 120 K.

NA MIRA DO RIFLE

Projetado para aumentar o realismo de jogos do tipo tiro ao alvo nos micros, o rifle óptico combina a aparência de uma arma com um sistema semelhante ao de uma câmara. Assim, o usuário pode dispensar o joystick.

O principal componente do rifle óptico da Stack (SLR, Stack Light Rifle) é a pistola de alvo eletrônico conectada ao computador por um longo cabo condutor. No terminal do computador, dependendo da versão, há um conector para o encaixe adequado.

Na versão do rifle para o micro ZX da Spectrum, o conector contém dois chips e alguns componentes simples para ligar a parte eletrônica interna da arma ao computador.

A pistola é provida de uma elegante coronha de ombro que se prende em sua parte traseira, um cano de rifle e uma imitação de mira telescópica, complementos que aumentam muito a precisão do tiro.

A parte eletrônica da pistola consiste num detector de luz ou fotodiodo, um pequeno amplificador e um buffer. A luz que vem pelo cano do rifle é focalizada por uma lente plástica sobre o fotodiodo, e o dispositivo detecta mudanças na intensidade da imagem. Depois de passar pelo amplificador, o sinal (uma forma ondulatória analógica) transforma-se num pulso digital transmitido ao computador quando se pressiona o gatilho.

A posição da tela observada no momento é aquela para a qual o rifle está apontando. Assim que recebe o pulso do SLR, o computador compara a posição do rifle naquele instante com a posição do alvo na tela. Se forem correspondentes, o jogador marca o ponto.

Disparo óptico

Existem variações do rifle óptico para o ZX da Spectrum, o Vic-20 da Commodore, o Commodore 64 e compatíveis. Todas desempenham as mesmas funções. Quanto ao software, a Stack fornece apenas três jogos em cassete com o SLR. Várias empresas de desenvolvimento de software fazem jogos apropriados a esse tipo de dispositivo, mas poucas produziram ou converteram programas para funcionar com ele.

A Stack não fornece programas utilitários que permitam ao usuário escrever seus próprios programas, nem divulga informações técnicas detalhadas sobre o funcionamento do rifle.

O SLR baseia-se no mesmo princípio de operação da caneta óptica. Contudo, é muito maior

O bamba do gatilho

e seu projeto prevê que seja mantido a uns 3 m da televisão, e não que esteja em contato com a tela. Para ajudar na filtragem da luz do ambiente, o SLR é provido de um tubo escuro (o cano da arma) e uma lente. Estes se combinam para alcançar um alto grau de precisão e permitem ao usuário atirar estando confortavelmente sentado numa poltrona. Os jogos disponíveis no mercado são, na verdade, exemplos pobres do que seria possível fazer; o uso de gráficos e a lógica e a ação dos jogos não sobressaem.

Ação congelada

Um dos maiores problemas na programação de canetas ópticas ou versões gigantes como o SLR é que os programas precisam ser muito eficientes. Em todos os cartuchos fornecidos pela Stack, os jogos param por um momento quando o gatilho é acionado. Isso porque, se a tela fosse varrida continuamente, como em geral ocorre com a caneta óptica, a velocidade do jogo diminuiria muito. Portanto, quando se aciona o gatilho, o software congela a ação para determinar se o alvo (na tela) está alinhado com a posição da arma. Uma vez que o software tenha determinado se o jogador acertou ou não o alvo, o jogo pode continuar.

Teoricamente, quando o gatilho é acionado, a quantidade de instruções necessárias para estabelecer a posição da tela em relação à posição detectada pela arma deveria ser muito pequena. Mas, observando-se o software em ação, percebe-se que nem sempre é esse o caso: às vezes a quantidade exigida é bem grande.

A provisão de recursos para a caneta óptica dentro do chip de vídeo tornaria a tarefa do software muito mais simples. O Commodore 64 oferece tal sistema, mas o ZX Spectrum não possui esses recursos, e a deficiência aparece quando se trata de calcular a posição do rifle, após o acionamento do gatilho.

Um tiro no escuro

A fotocélula detecta um ponto luminoso que se move varrendo a tela continuamente. O software tem o controle contínuo das coordenadas do ponto na tela; quando o rifle atira, o software pode comparar o valor das coordenadas com a posição do alvo para o qual o rifle estava apontado.

Temporada de caça

High Noon (Matar ou Morrer, também nome de um filme exibido no Brasil) é o melhor dos três programas de demonstração da Stack. A animação é boa e o pistoleiro da tela atira no jogador de modo verossímil. O Grouse Shoot (Tiro ao Faisão) e o Shooting Gallery (Tiro ao Alvo) apresentam um único alvo móvel, que precisa ser atingido antes que saia dos limites da tela. A animação é irregular e o acionamento do gatilho causa notável parada na tela.

SELEÇÃO ALEATÓRIA

Os arquivos de acesso aleatório, mais rápidos de manipular que os seqüenciais, exigem maior espaço de armazenamento, além de estruturação cuidadosa e rigorosamente uniforme.

As limitações dos arquivos seqüenciais devem-se à obrigatoriedade de ler as informações na ordem em que foram gravadas. Os arquivos de acesso aleatório ou direto não apresentam tais limitações porque os registros são acessados em qualquer ordem e com muita rapidez. A palavra "aleatório" não implica construção ou uso dos arquivos de maneira caótica. Significa apenas que qualquer segmento pode ser gravado ou lido sem a necessidade de passar pelas informações precedentes.

Os arquivos mantidos em fitas cassete, no entanto, são seqüenciais. Assim, não há possibilidade de acesso direto a um item de dados. O único modo de usar o acesso aleatório, nesse caso, consiste em carregar todos os dados na memória, mas isso limita o tamanho do arquivo. É preciso dispor de unidades de disco para se conseguir acesso aleatório útil; mesmo assim, algumas marcas de unidades de disco não admitem esse tipo de manipulação de arquivo.

Acesso aleatório x seqüencial

	ARQUIVOS DE ACESSO ALEATÓRIO	ARQUIVOS SEQÜENCIAIS
PRÓS	• Acesso rápido a registros específicos	• Conservam espaço • Disponíveis para sistemas de fitas
CONTRAS	• Desperdício de espaço • Exigem discos	• Lentos
APLICAÇÕES PARA AS QUAIS SÃO ADEQUADOS	• Dados previsíveis que estejam em formato definido • Quando pequenas quantidades de registros diferentes são acessadas; por exemplo, numa biblioteca onde os usuários solicitam detalhes sobre determinados livros. Esse é um aplicativo de baixa taxa de acerto.	• Grandes quantidades de dados não estruturados • Quando a maioria dos registros de um arquivo é processada; por exemplo, num sistema de folha de pagamento, em que cada empregado deve ser pago. Esse é considerado de alta taxa de acerto.

Os arquivos de acesso aleatório devem ser divididos em registros e campos. Para acessar o arquivo, especifica-se o registro requisitado, que, juntamente com os campos, será colocado num buffer na memória do computador. Aí, os campos podem ser apagados, corrigidos ou impressos. As estruturas mais complexas ficam a cargo do sistema operacional, que em curto tempo localiza o início de determinado registro no disco. Para facilitar a rápida localização, os registros têm o mesmo comprimento. Se cada um tiver 100 bytes de caracteres de comprimento e o programa receber instruções para gravar o número 83, o sistema operacional posicionará a cabeça do disco no início do byte de n.º 8.300 do arquivo. Ele possui um registro de quantos bytes estão em cada setor do disco, o que lhe permite calcular a posição do registro procurado. Esse método de leitura de arquivo pode parecer complexo e lento, mas é muito mais rápido que num arquivo seqüencial.

Ao padronizar os arquivos, é necessário escolher o tamanho que acomodará o mais longo dos registros. Os menores são preenchidos, em geral, com espaços (32 em código ASCII). Esse é o maior problema dos arquivos aleatórios, pois o preenchimento necessário para deixar o registro no tamanho escolhido acarreta um desperdício de valioso espaço de armazenamento. Por isso, reservam-se os arquivos aleatórios para pequenas quantidades de informação, quando o acesso precisa ser muito rápido, e deixam-se os arquivos seqüenciais para o armazenamento em massa.

Também os campos dentro de um registro devem ter dimensão padronizada, em especial nos sistemas que oferecem o recurso de acesso aleatório para campos e registros específicos. Mesmo nos sistemas sem tal recurso, esse procedimento constitui um modo mais organizado e eficiente de definição de arquivos. Quando se projeta um arquivo de acesso aleatório, o primeiro passo consiste em relacionar os diferentes campos e escolher tamanhos apropriados para eles. Por exemplo, o campo para o nome de uma pessoa compreende pelo menos vinte caracteres de comprimento, enquanto para armazenar sua idade bastam dois.

A economia é essencial ao se projetar um arquivo, pois sempre haverá intercâmbio entre o total de informações armazenadas e o número de registros diferentes. É freqüente que os sistemas de codificação sejam projetados para reduzir o espaço tomado pelos dados. Por exemplo, os códigos 1, 2 e 3, para preto, vermelho e verde, ou códigos de dados, como 841011, para 11

de outubro de 1984. No entanto, os sistemas de codificação devem permanecer internos ao sistema, e os programas precisam reconverter os códigos para uma forma de fácil compreensão, quando um campo é introduzido ou apresentado na tela.

Comprimento dos arquivos

A maioria dos sistemas limita o comprimento disponível — pode variar de 128 bytes a até 2.048 bytes. Além disso, muitas vezes é mais eficiente escolher um tamanho múltiplo do setor — usam-se, em geral, números como 64, 128, 256 ou 512.

Evita-se, assim, que os registros individuais se dividam, abrangendo mais de um setor — ou seja, reduz-se drasticamente o número de acessos ao disco.

A manipulação dos arquivos aleatórios costuma ser bem mais simples que a dos arquivos seqüenciais. Em ambos os sistemas, é necessário manter uma contagem atualizada do número de registros; e muitas vezes, nos arquivos de acesso aleatório, usa-se o primeiro registro (em geral, o de n.º 0) para armazenar essa e outras informações relevantes, como a data de criação do arquivo. A rígida estrutura do campo e do registro seria descartada para esse registro.

A leitura do registro é feita pelo número, segundo técnicas semelhantes às usadas para pesquisar dados de matrizes em BASIC. Muitas vezes, usa-se como chave para o arquivo um campo específico, como, por exemplo, o campo do nome. O computador lê o campo-chave e monta um índice que identifica onde estão armazenados os vários nomes.

Os arquivos aleatórios não indexados costumam ser acessados registro por registro, como os arquivos seqüenciais. Mas, se os registros estão ordenados pelo campo-chave, podem-se usar métodos rápidos de busca. Imagine que se pretenda encontrar "João" num arquivo ordenado pelo nome. Começa-se a procura no meio do arquivo e descobre-se, por exemplo, que o nome aí registrado é "Paulo", que vem depois de "João", na ordem alfabética. Assim, elimina-se o restante do arquivo. O palpite seguinte será um registro no meio dessa primeira metade do arquivo. O nome poderá ser "Homero", exigindo novo avanço, e assim por diante. Tais técnicas tornam-se, às vezes, bastante sofisticadas. O desempenho de muitos programas aumenta quando se mantêm na RAM os registros mais usados. Como resultado, os registros são localizados e armazenados em arquivos grandes a alta velocidade.

Eliminar e inserir novos registros pode ser relativamente lento. O método mais simples de eliminar consiste em copiar o registro seguinte em seu espaço, sobrepondo as informações nele contidas. Cada registro subseqüente é copiado uma posição antes e, afinal, a contagem de registros fica com um a menos. De modo semelhante, insere-se novo registro em qualquer ponto movendo-se o último registro um número para a frente e copiando um espaço à frente todos os registros entre ele e o número do novo registro. Isso cria um espaço em que o novo registro pode ser gravado.

Nenhuma dessas técnicas é rápida, embora todas sejam mais eficientes que operações semelhantes com arquivos seqüenciais. As inserções e eliminações de registros ficam mais rápidas se o arquivo tiver um índice separado. Marca-se no índice que o registro foi eliminado. Os dados permanecem inalterados. À medida que se acrescentam novos registros, eles podem ser encaixados em arquivos não utilizados ou eliminados e o índice é atualizado.

Há duas vantagens ainda a considerar no sistema de arquivo de acesso aleatório. Em primeiro lugar, embora seja mais rápido ler e gravar grupos de registros juntos, os arquivos podem sair da ordem. A maioria dos programas oferece um recurso de ordenação que organiza os registros em ordem lógica e apaga os registros eliminados. Em segundo lugar, o sistema de marcar registros eliminados oferece segurança, pois permite recuperar esses dados, se necessário. A segurança existirá até que os registros eliminados sejam sobrepostos ou descartados por um programa de ordenação.

Um novo registro
Estas duas técnicas para inserir e eliminar registros de um arquivo de acesso aleatório são simples mas eficientes. Para inserir, deslocam-se os registros um número de registro abaixo. Para eliminar, deslocam-se os registros para cima, sobrepondo o registro a ser descartado.
Técnicas mais avançadas empregam um índice para os novos registros atuais e os eliminados, evitando, assim, a cópia e o deslocamento dos registros, que consomem muito tempo.

MAGIA ANIMAL

Neste programa, o computador descobre o animal em que o usuário está pensando. O jogo usa princípios da inteligência artificial e dá à máquina uma aparente capacidade de raciocinar e adquirir conhecimentos.

Este é um jogo em que o computador tenta adivinhar o nome do animal escolhido pelo jogador. Ele o faz por meio de perguntas do tipo "voa?", "tem pêlos?" etc. Você só pode responder "sim" ou "não" e o computador vai ponderando esses dados até se sentir em condições de fazer uma tentativa abalizada. É surpreendente, sobretudo para pessoas não familiarizadas com a informática e com a heurística, que o programa possa fazer isso.

Dois aspectos tornam o programa muito divertido: a capacidade do computador de se comunicar em linguagem compreensível (embora as próprias respostas do usuário a suas perguntas sejam limitadas a sim e não) e o estoque de conhecimento a que o computador pode recorrer para adivinhar o animal.

Aprendendo a jogar

Magia Animal é um programa heurístico — ensina a si mesmo a melhorar seu desempenho enquanto vai sendo executado.

Quando usado pela primeira vez, o programa "conhece" somente dois nomes de animais e uma única pergunta. Conforme você responda sim ou não, ele tenta adivinhar a resposta. Quando o computador faz uma tentativa errada, o programa pede que você dê o nome do animal escolhido e uma pergunta para distingui-lo do palpite que o programa deu.

Essas informações são então acrescentadas ao banco de dados do programa para construir uma árvore de opções, que ele poderá usar no jogo seguinte. Toda vez que você utiliza o jogo, a árvore aumenta de tamanho, até que, afinal, o programa vai acertar a maioria dos animais.

Ainda assim, o programa *não sabe* nada sobre animais. Está seguindo às cegas um guia constituído pelo conhecimento combinado de todos os jogadores que o usaram. As informações poderiam muito bem ser a respeito de diferentes tipos de cerveja, nomes de escritores, peças de motocicleta, sintomas de doenças ou amigos e parentes do jogador. Uma versão do programa que possibilitasse definir a pergunta inicial e duas respostas poderia ser usada em várias tarefas diferentes. Não são os dados que fazem o programa funcionar, mas o modo como vão sendo organizados.

É fácil construir a árvore com um simples programa em linguagem BASIC. A maioria das estruturas desse tipo é mantida em matrizes; no caso, usam-se T$() para as questões e os nomes dos animais e "sim" e "não" para as ligações entre as entradas específicas em T$. Essas ligações formam o trajeto até a resposta.

Para qualquer entrada em T$, o elemento correspondente em S() informa o programa sobre onde buscar, se a resposta à questão for sim. De modo semelhante, o elemento N() é a ligação para a resposta negativa. No final da árvore, o texto em T$() não é uma pergunta, mas o nome de um animal. Nesse caso, tanto S() como N() são colocados em 0 e o programa precisa adivinhar em que o jogador está pensando.

Nossa versão do programa, curta e simples, mostra os princípios envolvidos. Se quiser aperfeiçoá-la, você pode melhorar a apresentação para sua máquina pelo acréscimo de gráficos em cores, de som etc.

Variações

Para a linha Sinclair

Esse programa foi escrito em BASIC para os compatíveis com Apple, podendo ser convertido para a maioria dos micros com adaptações mínimas.

Para a linha Sinclair, as instruções devem vir em linhas separadas, o que implica alterar os GOTOs e os GOSUBs. Todas as atribuições de valor devem ser precedidas pela palavra LET. Outras alterações que você deve fazer:

Inclua estas duas linhas:

```
45 LET L=40
55 T$ (N,L)
```

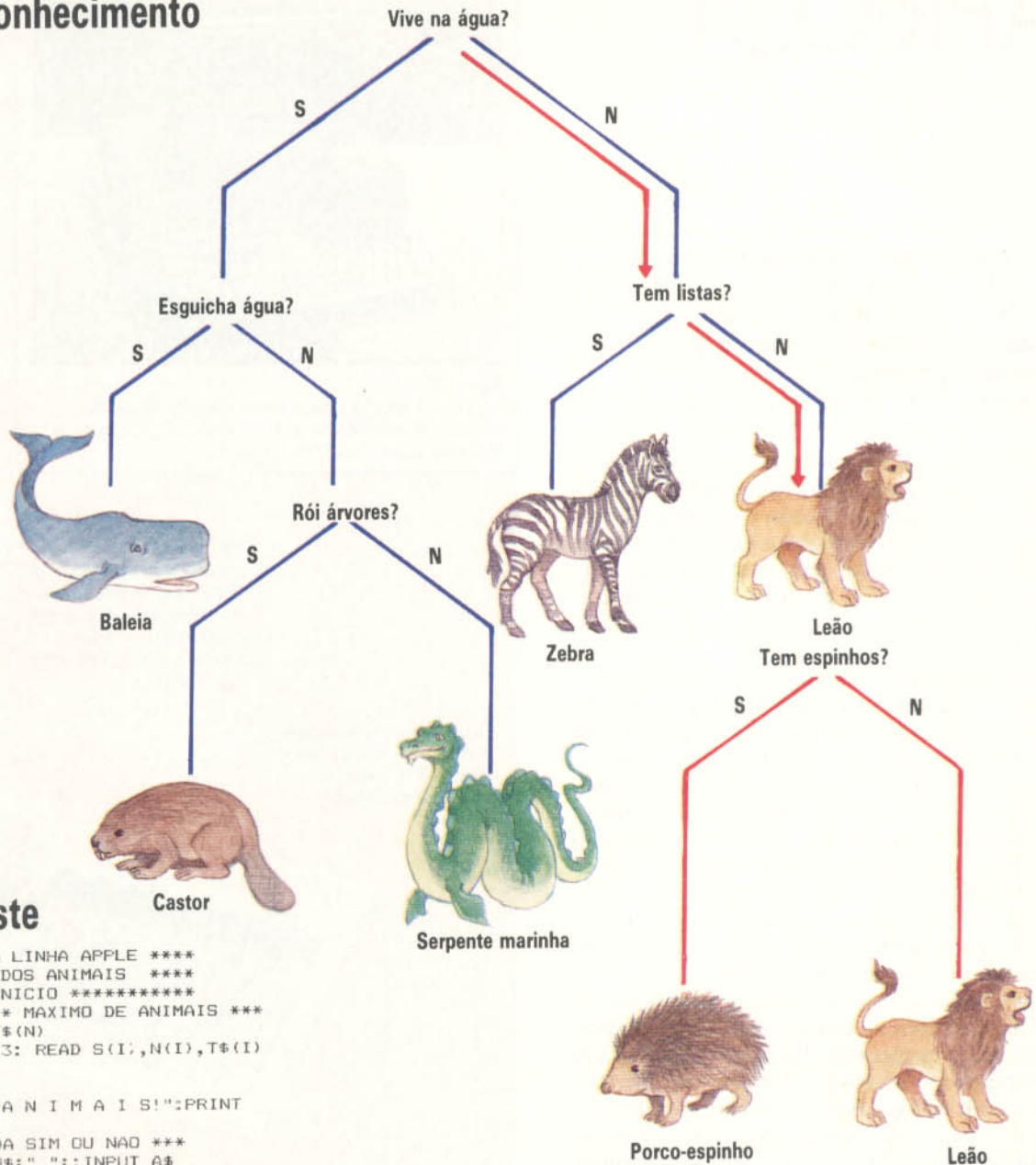
Na linha 200, substitua END por STOP.
Escreva a linha 230 deste modo:

```
230 IF Y(P)=0 AND N(P)=0 THEN GOTO
    290
```

A árvore do conhecimento

O diagrama mostra a árvore de Magia Animal depois que vários jogos foram executados. O computador aprendeu a reconhecer cinco animais diferentes, com quatro perguntas para possibilitar a distinção entre eles. No fluxo em vermelho *(à direita)*, você percebe como o computador usa a árvore para responder às perguntas do jogador na próxima partida. Desta vez, o jogador está pensando num porco-espinho e o computador é informado de que não se trata de um leão. Solicita então um modo de distinguir entre porco-espinho e leão, para que ele possa reconhecer o novo animal.

Quando você tiver sua versão do programa, inclua um camelo.

Vive na água?
S
N
Esguicha água?
S
N
Baleia
Rói árvores?
S
N
Castor
Serpente marinha
Tem listas?
S
N
Zebra
Leão
Tem espinhos?
S
N
Porco-espinho
Leão

Programa-teste

```
10 REM **** PARA A LINHA APPLE ****
20 REM ****  JOGO DOS ANIMAIS  ****
30 REM ********* INICIO ***********
40 N = 100: REM *** MAXIMO DE ANIMAIS ***
50 DIM S(N),N(N),T$(N)
60 C = 3: FOR I = 3: READ S(I;,N(I),T$(I)
   :NEXT I
65 HOME
70 PRINT : PRINT "A N I M A I S!":PRINT
80 GOTO 190
90 REM *** RESPONDA SIM OU NAO ***
100 PRINT :PRINT Q$;" ";:INPUT A$
110 IF A$ = "SIM" OR A$="S" THEN A = 1:
    RETURN
120 IF A$ = "NAO" OR A$="N" THEN A = 0:
    RETURN
130 PRINT : PRINT " RESPONDA SIM OU
    NAO": GOTO 100
180 REM *** INICIA UM NOVO JOGO ***
190 Q$ = "QUER JOGAR":GOSUB 100
200 IF A = 0 THEN PRINT :PRINT "TCHAU !!"
    : END
210 P = 1
220 REM *** ROTINA PRINCIPAL ***
230 IF S(P) = 0 AND N(P) = 0 THEN 290
240 Q$ = T$(P): GOSUB 100
250 IF A = 1 THEN P = S(P)
260 IF A = 0 THEN P = N(P)
270 GOTO 230
280 REM *** ADIVINHA QUAL E' O ANIMAL ***
290 A$ = T$(P): T$ = A$
300 Q$="E' " + A$: GOSUB 100
310 IF A = 1 THEN PRINT " ACERTEI!!!":
    GOTO 430
320 REM *** APRENDE OUTRO ANIMAL ***
330 PRINT : PRINT " DESISTO!!!":PRINT
    "QUAL E' O ANIMAL ";:INPUT N$
340 A$ = N$
350 PRINT : PRINT "PROPONHA UMA PERGUNTA
    QUE FACA A DISTINCAO ENTRE ";A$;" E "
    ;T$: INPUT D$
360 Q$ ="QUAL A RESPOSTA PARA" + T$:GOSUB
    100
370 A$= T$(P): T$(P) = D$: T$ (C+1) = A$:
    T$(C+2) = N$
380 IF A = 1 THEN S(P) = C + 1: N(P) = C
    + 2
390 IF A = 0 THEN S(P) = C + 2: N(P) = C
    + 1
400 S(C + 1) = 0: N(C+1) = 0:S
    (C+2)=0:N(C+2)=0
410 C = C + 2
420 REM *** FIM DO JOGO & LOOP PARA NOVA
    PARTIDA ***
430 A = INT (C/2) + 1
440 PRINT : PRINT "AGORA JA' SEI ";A;"
    ANIMAIS!"
450 GOTO 190
460 REM *** DADOS INICIAIS ***
470 DATA 2,3,"VIVE NA AGUA"
480 DATA 0,0,"A BALEIA"
490 DATA 0,0,"O LEAO"
```

APRICOT

Compacto, mas não verdadeiramente portátil, o Apricot segue a tendência atual de apresentar o gabinete da CPU, o vídeo e o teclado separados. Para amadores, é uma máquina cara.

Com inúmeras características projetadas para atrair o profissional, o microcomputador inglês Apricot destaca-se sobretudo pelo design do teclado e do monitor. Versátil e inovador, o teclado do Apricot conta com o recurso extra da Microtela — um visor de cristal líquido de 2 linhas e 40 colunas — e mais seis teclas programáveis pelo usuário.

Quando se liga a máquina, um programa-teste mostra a quantidade de memória disponível (256 Kbytes é o padrão, mas pode ser ampliado para 768 Kbytes) e solicita ao usuário que insira o disco mestre do MS-DOS. Para usuários não familiarizados com sistemas operacionais como o CP/M ou o MS-DOS, um menu interativo — o Manager ("gerenciador") — possibilita a fácil escolha de softwares aplicativos (incluindo o Supercalc, o Multiplan e o BASIC da Microsoft) ou utilitários (a exemplo do configurador do teclado ou do gerador de caracteres).

As funções atribuídas às teclas programáveis podem ser apresentadas no visor LCD. Assim, os menus de opções exibidos na tela principal são duplicados na Microtela. Tocar a tecla de função apropriada equivale a selecionar o item no menu de tela com o uso das teclas de movimentação de cursor e [Return]. O único inconveniente encontra-se nas teclas tipo membrana, sensíveis ao toque — menos eficientes do que as do tipo máquina de escrever.

Há também as oito teclas habituais de função, marcadas com os nomes de suas funções normais — HELP, PRINT, MENU, FINISH, entre outras. Como as demais teclas do Apricot, essas podem ser reconfiguradas por meio do programa Keyedit (Editor de Teclas) que acompanha a máquina.

O software que acompanha o Apricot consiste na planilha financeira Supercalc e num conjunto amplo de utilitários.

O fabricante fornece poucas informações sobre o hardware, embora os utilitários que o acompanham sejam suficientes para começar a trabalhar o computador. Não há detalhes sobre o mapeamento da memória nem sobre as chamadas do sistema, que poderiam ser necessários ao desenvolvimento de novos softwares.

Apricot XI
Em vista da reduzida capacidade de armazenamento dos microdiscos, a versão Apricot XI incorpora um disco rígido de 10 Mbytes e uma unidade de microdisco.

Tela
Um monitor de fósfo
alta resolução forne
imagem clara e bem

LCD
Um visor de cristal líquido, com 2 linhas, que pode ser usado para mensagens, como relógio, ou calculadora. O relógio permanece funcionando com o micro desligado.

Microdiscos Sony
Dois microdiscos de 315 K permitem o armazenamento compacto e apropriado.

Teclas de função interativas
Proporcionam funções padrão do tipo HELP e REPEAT para programas aplicativos.

Teclas de função tipo membrana
Estas seis teclas podem ser programadas segundo as informações exibidas no visor de cristal líquido, para operar com programas específicos.

RAM com 256 K
Esta ampla memó
no equipamento.

O monitor do Apricot
Uma característica original do Apricot é o monitor de vídeo deslocável em três direções: para os lados, deslizando sobre o gabinete da CPU; em rotação sobre o seu pedestal; e inclinando-se para a frente e para trás, de modo a facilitar a visão sob diferentes condições de iluminação. Apesar das 9" de largura, a tela permite mostrar 132 caracteres de largura por cinqüenta de altura.

Process
Um seg
é usado
funções
(input/o

ro verde de
ce uma
definida.

Saída serial RS232
Conexão serial para impressoras, modems e outros periféricos.

Saída paralela
Conexão para saída paralela padrão Centronics.

ROM
A ROM do Apricot contém programas de inicialização e de autoteste.

Slots para expansão
Dois slots para expansão admitem placas extras, tais como memória adicional e um modem.

ria é padrão

ador de I/O
ndo microprocessador
para manipular as
de entrada e saída
tput).

CPU
Microprocessador 8086 de 16 bits.

Encaixe para o 8087
Há espaço para um co-processador aritmético 8087 destinado a cálculos numéricos em alta velocidade.

APRICOT

MICROPROCESSADOR

8086, com opção para processador aritmético 8087.

MEMÓRIA

256 K de RAM, expansível para 768 K.

SISTEMAS OPERACIONAIS

MS-DOS, CP/M-86 e CP/M-86 Concorrente.

VÍDEO

Monitor monocromático, com resolução gráfica de 800 x 400 pixels. No modo texto, a tela apresenta 132 x 50 caracteres, ou 80 x 25. O monitor permite três tipos de movimento, para maior visibilidade.

TECLADO

Destacável, padrão IBM PC, com noventa teclas, mais seis do tipo membrana, programáveis para funções definidas pelo LCD.

ARMAZENAMENTO DE DADOS

O modelo padrão tem duas unidades para microdiscos Sony de 3 1/2", com 315 ou 720 K de capacidade cada um. Há outra versão do Apricot que possui um disco rígido de 10 Mbytes e uma só unidade para microdisco.

INTERFACES

Conectores para saída paralela, serial e mouse. Dois slots para placas opcionais ou modem.

DOCUMENTAÇÃO

Ampla e bem apresentada.

A LÓGICA DA PROGRAMAÇÃO

As operações lógicas E e OU são inestimáveis instrumentos de programação. A maioria dos conjuntos de instrução em linguagem BASIC ou código de máquina inclui E e OU entre seus comandos.

Os dois operadores lógicos E (AND) e OU (OR) têm vários usos; o mais comum relaciona dois ou mais enunciados condicionais. Tente prever o resultado deste programa BASIC:

```
10 FOR I=1 TO 5
20 FOR J=1 TO 5
30 IF I=3 AND J=2 THEN PRINT I,J
40 NEXT J
50 NEXT I
60 END
```

O programa rodará por meio do par de loops, mas imprimirá os valores de I e de J somente se I=3 e J=2. Ou seja, na tela aparecerá o seguinte resultado:

```
2
```

A operação lógica OU pode ser usada de modo muito semelhante. Mudando a linha 30 para:

```
30 IF I=3 AND J=2 OR J=4 THEN PRINT I,J
```

será produzida a saída

```
14
2 4
3 2
3 4
4 4
5 4
```

O computador executa a operação E com prioridade em relação à operação OU: I e J serão impressos se I=3 e J=2 ou se J=4. A ordem de prioridade pode ser modificada pelo uso de parênteses. Verifique qual será o resultado do programa se a linha 30 for mudada para

```
30 IF I=3 AND (J=2 OR J=4) THEN PRINT I,J
```

Muitos micros usam registros especiais para controlar várias funções da máquina. Cada bit dentro de registros desse tipo pode controlar um aspecto diferente da operação. Por exemplo, no Commodore 64 há um registro de 8 bits que ativa e desativa os sprites. Cada bit no registro se relaciona com um dos oito sprites disponíveis. Se qualquer bit no registro for colocado em um (1), o sprite que ele controla ficará visível na tela. Se o bit for colocado em zero (0), o sprite será desativado.

Com a utilização do BASIC torna-se fácil ativar qualquer combinação de sprites, calculando o número binário exigido de 8 bits e armazenando pelo comando POKE seu equivalente decimal no registro. Esse método, porém, não leva em conta o estado do registro anterior ao comando POKE e pode resultar na desativação de sprites anteriormente ativados. A solução para esse problema está no desenvolvimento de uma técnica que permita ao programador isolar os bits a serem mudados sem alterar qualquer um dos outros.

Para demonstrar essa técnica, suponha que os sprites 0, 1, 5 e 6 estejam ativados. O registro que ativa terá a forma

Número do sprite	7	6	5	4	3	2	1	0
Bit correspondente	0	1	1	0	0	0	1	1

Agora ative o sprite 4, lendo o registro com o PEEK, operando com OR os conteúdos com 16 (00010000) e obtendo o resultado com o POKE.

Byte original	0	1	1	0	0	0	1	1
Byte operado com OU	0	0	0	1	0	0	0	0
Dá o novo byte	0	1	1	1	0	0	1	1

Com o comando BASIC POKE reg, PEEK (reg) OR 16, pode-se ativar o bit 4. Para desativá-lo, é preciso acessar o registro com PEEK e operar com AND seu conteúdo, usando o número 239.

Novo byte	0	1	1	1	0	0	1	1
E	1	1	1	0	1	1	1	1
Byte original	0	1	1	0	0	0	1	1

Observe que o número 239 é obtido subtraindo-se 16 de 255. Usando o comando BASIC POKE reg, PEEK (reg) AND 239, restaura-se o registro em seu estado original.

Essas técnicas são amplamente usadas na programação em código de máquina, na qual a alteração do estado dos registros de controle pode formar parte importante do programa.

CENAS ANIMADAS

O sprite é uma forma gráfica definida pelo usuário, que pode ser movida na tela à velocidade especificada pelo computador. A combinação de sprites permite compor cenas animadas.

Respostas do exercício anterior

1a) $A \cdot (\bar{A}+\bar{B}) =$
$= A \cdot \bar{A} + A \cdot \bar{B}$ (lei distributiva)
$= A \cdot \bar{B}$ $(A \cdot \bar{A} = 0)$

b) $X + Y \cdot (X+Y) + X \cdot (\bar{X}+Y) =$
$= X + Y + X \cdot (\bar{X}+Y)$ (relação 5)
$= X + Y + X \cdot Y$ (relação 6)
$= X + Y$ (absorção)

c) $P \cdot Q + \bar{P} \cdot Q + \bar{P} \cdot \bar{Q} =$
$= P \cdot Q + \bar{P} \cdot (Q + \bar{Q})$ (lei distributiva)
$= P \cdot Q + \bar{P}$ $(Q + \bar{Q} = 1)$
$= \bar{P} + Q$ (dual da relação 6)

d) $\overline{\bar{X} + \bar{Y} \cdot Z + \bar{Z} \cdot Y} =$
$= \bar{\bar{X}} \cdot \overline{\bar{Y} \cdot Z} \cdot \overline{\bar{Z} \cdot Y}$ (de Morgan)
$= X \cdot Y \cdot Z \cdot (Z + \bar{Y})$ $(\bar{\bar{X}} = X$, de Morgan)
$= X \cdot Y \cdot Z \cdot Z + X \cdot Y \cdot Z \cdot \bar{Y}$ (lei distributiva)
$= X \cdot Y \cdot Z + 0$ $(Z \cdot Z = Z,\ Y \cdot \bar{Y} = 0)$
$= X \cdot Y \cdot Z$

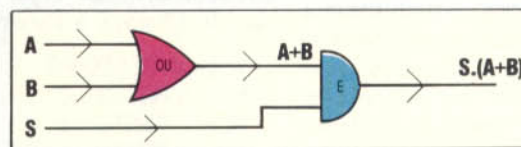
2) A tabela de validação do sistema de alarma é:

ENTRADAS			SAÍDAS
A	B	S	Alarma
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

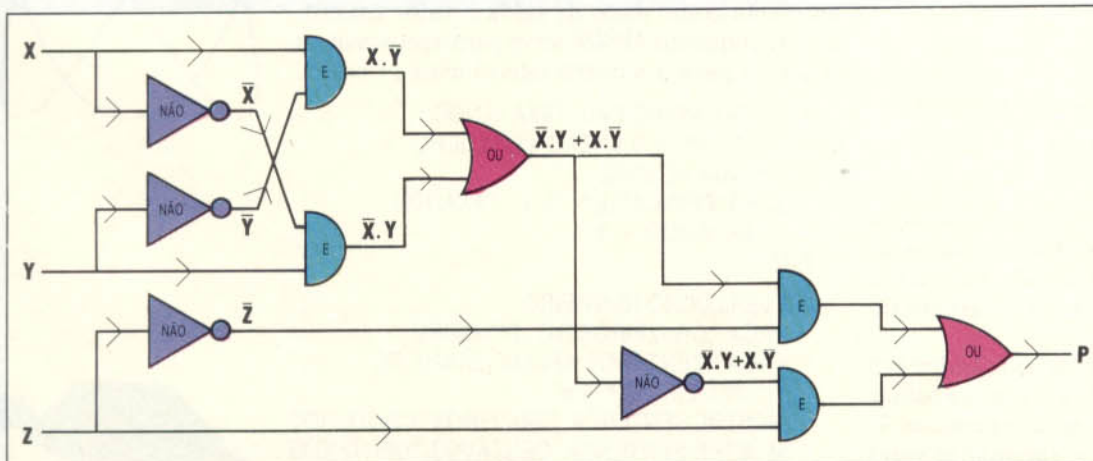
Alarma $= \bar{A} \cdot B \cdot S + A \cdot \bar{B} \cdot S + A \cdot B \cdot S$
$= \bar{A} \cdot B \cdot S + A \cdot S \cdot (\bar{B} + B)$ (lei distributiva)
$= \bar{A} \cdot B \cdot S + A \cdot S$ $(\bar{B} + B = 1)$
$= S \cdot (A + \bar{A} \cdot B)$ (lei distributiva)
$= S \cdot (A + B)$ (dual da relação 6)

3) Se os três interruptores forem X, Y e Z e a luz for P, a tabela verdade será:

ENTRADAS			SAÍDAS
X	Y	Z	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$P = \bar{X} \cdot \bar{Y} \cdot Z + \bar{X} \cdot Y \cdot \bar{Z} + X \cdot \bar{Y} \cdot \bar{Z} + X \cdot Y \cdot Z$
$= Z \cdot (\bar{X} \cdot \bar{Y} + X \cdot Y) + \bar{Z} \cdot (\bar{X} \cdot Y + X \cdot \bar{Y})$ (lei distributiva)
$= Z \cdot (\overline{\bar{X} \cdot Y + X \cdot \bar{Y}}) + \bar{Z} \cdot (\bar{X} \cdot Y + X \cdot \bar{Y})$ (de Morgan)

4) A pergunta "Você fala a verdade?" é pouco útil, pois tanto o mentiroso como o que fala a verdade darão a mesma resposta. A tabela de validação tem a mesma forma que a função $X \cdot Y + \bar{X} \cdot Y$, que pode ser simplificada para Y. Isto é, a resposta depende de uma variável, não de duas, o que não permite diferenciar os mentirosos dos sinceros. Mas, se fizermos a pergunta "Porcos têm asas?", a tabela será:

		RESPOSTAS POSSÍVEIS	
		SIM	NÃO
IDENTIDADE POSSÍVEL DO QUE RESPONDE	MENTIROSO	1	0
	SINCERO	0	1

e essa é a tabela de validação para a função $X \cdot \bar{Y} + \bar{X} \cdot Y$, que é também uma tabela para OU-exclusivo. Essa pergunta permite identificar aquele que responde.

CALCULANDO COM O LOGO

Se você precisa realizar muitos cálculos numéricos, dificilmente a linguagem LOGO será sua primeira opção. Apesar disso, ela dispõe de surpreendentes recursos para o processamento de números.

Quase todas as versões do LOGO possibilitam a realização de operações aritméticas seja com números reais (decimais), seja com números inteiros. Para isso são utilizados os operadores infixos: +, −, * e /. Esses operadores denominam-se "infixos" porque são escritos entre os números que relacionam — por exemplo, 3 + 4. Algumas versões do LOGO também operam com a notação por prefixos, segundo a qual o exemplo acima seria reescrito como SOMA 3 4. Essa notação por prefixos possui a vantagem de ser compatível com a forma na qual são escritos os outros comandos e operações do LOGO.

O MLOGO utiliza apenas a notação por infixos. Contudo, permite a programação de comandos com prefixos, caso esta seja necessária. Defina e teste as rotinas SOMA e PRODUTO:

```
AP SOMA :A :B
  SAIDA :A + :B
FIM
AP PRODUTO :A :B
  SAIDA :A * :B
FIM
```

A "precedência" das operações (a ordem na qual elas são realizadas) segue as regras usuais da matemática. Todas as operações indicadas entre parênteses são efetuadas em primeiro lugar, seguidas pelas multiplicações e divisões e, por fim, pelas adições e subtrações:

```
MOSTRE (3 + 4) * 5
MOSTRE 3 + 4 * 5
```

Experimente agora a notação por prefixos:

```
MOSTRE PRODUTO 5 SOMA 3 4
MOSTRE SOMA 3 PRODUTO 4 5
```

Isso demonstra uma outra vantagem da notação por prefixos — não há necessidade de regras de precedência e a linha é considerada do mesmo modo que qualquer outra linha de comandos do LOGO.

A operação normal de divisão (/) produz um número real como resultado. Para se trabalhar com números inteiros, duas outras operações — QUOC (QUOCIENTE) e RESTO — são muitas vezes de grande utilidade.

```
QUOC 47 5 E 9
RESTO 47 5 E 2
```

Um método padronizado para se converter em binário um número decimal consiste em dividi-lo sucessivamente por dois até que o resultado seja zero. O número binário é obtido ao se escrever, em ordem inversa, os restos encontrados em cada divisão. Veja, por exemplo, a conversão de 12 para o sistema binário:

```
12/2 = 6; RESTO = 0
6/2 = 3; RESTO = 0
3/2 = 1; RESTO = 1
1/2 = 0; RESTO = 1
```

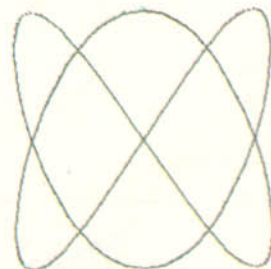
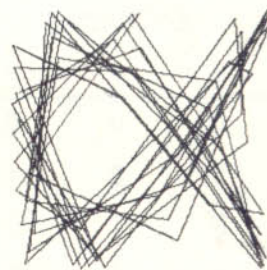
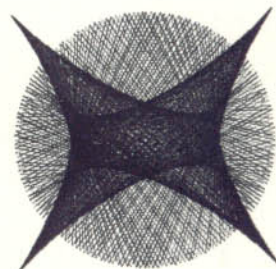
Assim, lendo-se os restos de baixo para cima, obtemos o número binário equivalente ao decimal 12, ou seja, 1.100.

Os comandos QUOC e RESTO facilitam bastante a implementação desta técnica no LOGO. Se colocarmos a instrução MOSTRE *depois* da chamada recursiva, os restos serão apresentados na ordem correta (inversa).

```
AP BINARIO :X
  SE :X = 0 ENTAO PARE
  BIN QUOC :X 2
  MOSTRAR RESTO :X 2
FIM
```

Existem duas operações para o arredondamento dos números — INT (INTEIRO) e APROX (APROXIMAR). INT apresenta como dado de saída a parte inteira de um número, ignorando todos os algarismos depois da vírgula. APROX aproxima o número para o inteiro mais próximo, seja este maior ou menor.

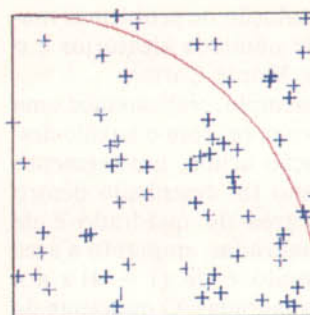
As rotinas apresentadas a seguir calculam os juros compostos sobre um investimento de acordo com determinada taxa. Em VER.LUCRO, INT apresenta como dado de saída o valor em cruzeiros, enquanto APROX serve para aproximar os centavos para o número inteiro mais próximo.

```
AP JURO :PRINCIPAL :TAXA :ANOS
  SE :ANOS = 0 ENTAO VER.LUCRO
  :PRINCIPAL PARE
  JURO :PRINCIPAL * (1 + :TAXA/100)
  :TAXA :ANOS - 1
FIM
AP VER.LUCRO :DINHEIRO
  FACA "CRUZEIROS INT :DINHEIRO
  FACA "CENTAVOS APROX (:DINHEIRO -
    :CRUZEIROS) * 100
  MOSTRE SENTENCA :CRUZEIROS "CRUZEIROS
  MOSTRE SENTENCA :CENTAVOS "CENTAVOS
FIM
```

Testes lógicos

Já empregamos =, < e > como testes lógicos em várias rotinas. As operações lógicas SETODOS, SEUM e NAO podem ainda ser combinadas de modo a constituir outros testes. SETODOS é verdadeiro apenas quando ambos os seus dados de entrada são verdadeiros. SEUM é verdadeiro quando um de seus dados de entrada for verdadeiro, enquanto NAO só é verdadeiro quando seu dado de entrada for falso. Desse modo obtemos:

```
SE SEUM :X>0 :X = 0 ENTAO MOSTRE "POSITIVO
SE NAO :X<0 ENTAO MOSTRE "POSITIVO
SE SETODOS :X>0 :X<100 ENTAO MOSTRE
  [ENTRE 0 E 100]
```

A operação NUMERO? tem como resultado VERD (VERDADEIRO) se o dado introduzido for um número. Caso contrário, o resultado será FALSO. Esta operação foi implementada na rotina PRIMO?, que apresenta como resultado VERD se o dado de entrada for um número primo, ou FALSO em caso contrário. Ela começa pela confirmação de que o dado introduzido é realmente um número, e de que este é maior do que dois. PRIMO.TESTE verifica a seguir se algum inteiro entre a raiz quadrada do número e o número dois pode dividi-lo sem deixar resto.

```
AP PRIMO? :NUM
  SE NAO NUMERO? :NUM ENTAO MOSTRE [ISTO
    NAO E UM NUMERO] PARE
  SE :NUM<2 ENTAO SAIDA "FALSO
  SAIDA PRIMO.TESTE :NUM INT RQD :NUM
FIM
```

MÉTODO DE MONTE CARLO

```
AP PRIMO.TESTE :NUM :FATO
  SE :FATO = 1 ENTAO SAIDA "VERD
  SE (RESTO :NUM :FATO) = 0 ENTAO SAIDA
    "FALSO
  SAIDA PRIMO.TESTE :NUM :FATO - 1
FIM
```

Números aleatórios

O resultado de SORTEIE n é um número inteiro aleatório entre 0 e n-1. A rotina BEBADO faz com que a tartaruga atravesse a tela cambaleando. A cada passo que dá, ela faz um giro de ângulo aleatório. O dado de entrada A determina o maior ângulo possível de giro. Se você executar essa rotina, verá que a tartaruga se move em círculos imprecisos, girando para a direita ou para a esquerda conforme o valor atribuído a A.

```
AP BEBADO :A
  FR 1
  DI (-:A/2 + SORTEIE :A)
  BEBADO :A
FIM
```

Um passo fora da linha

Segundo o teorema Passeio de Bêbado, após n passos em direções perfeitamente aleatórias, supera 0,5 a probabilidade de que a distância percorrida pelo bêbado desde o ponto inicial seja menor do que a raiz quadrada de n. Esta previsão estatística baseia-se num número muito alto de passos. Você pode verificá-la com o LOGO:

```
AP PASSEIOBEBADO :PASSONUM
 :PASSO
  LT REPITA :PASSONUM [DI
    (SORTEIE-361) FR :PASSO]
FIM
```

DRUNKARD'S WALK

Uma técnica para a resolução de problemas matemáticos por meio de números aleatórios é o chamado "método de Monte Carlo".

Em seguida, como exemplo, realizaremos uma aproximação ao número pi (π) com o auxílio desse método. Na ilustração acima, um segmento de um quarto de círculo foi desenhado dentro de um quadrado. A área do quadrado é de 100 x 100 unidades quadradas, enquanto a área de um quarto de círculo é de (1 ÷ 4) x π x x 100 x 100 unidades quadradas. O quociente da divisão da área do círculo pela do quadrado é igual a π ÷ 4. Agora, lance ao acaso um alfinete sobre o quadrado, repita isto 1.000 vezes e conte quantas vezes o alfinete cai dentro da área do quarto de círculo. Esse número será chamado de IN. O valor de IN/1.000 deve ser aproximadamente o mesmo que o resultado de: círculo ÷ quadrado — isto é, π ÷ 4. Assim, se você fizer essa experiência, multiplicar IN por 4 e dividir o resultado por 1.000, terá uma aproximação de π. Isso é o que as rotinas seguintes fazem:

```
AP MONTE.CARLO
  DESENHE
  SEMCOR
  FACA "IN 0
  MC1 1000 100 100
  MOSTRE SENTENCA [VALOR
    DE π E] 0.004 * :IN
FIM

AP MC1 :NUM :XNUM :YNUM
  SE :NUM = 0 ENTAO PARE
  SORTEIE.PONTO :XNUM YNUM
  SE DENTRO? ENTAO FACA "IN :IN + 1
  MC1 :NUM - 1 :XNUM :YNUM
FIM
```

A rotina MONTE.CARLO apenas estabelece as condições, chama MC1 e mostra os resultados. MC1 faz a maior parte do trabalho, chamando SORTEIE.PONTO para posicionar a tartaruga e aumentando o valor de IN se o ponto estiver dentro do círculo. Isso se repete até que a rotina tenha sido executada o número determinado de vezes.

```
AP SORTEIE.PONTO :XNUM :YNUM
  DEFXY SORTEIE :XNUM SORTEIE :YNUM
FIM

AP DENTRO?
  SE (CORX * CORX + CORY) * CORY) < 1000
    ENTAO SAIDA "VERD
  SAIDA "FALSO
FIM
```

SORTEIE.PONTO posiciona a tartaruga em um ponto aleatório dentro do quadrado. DENTRO? verifica se a tartaruga está dentro do círculo. Essa rotina leva algum tempo para ser rodada, mas no final será obtido um resultado aproximado para o número π.

As curvas de Lissajous formam uma família de curvas muito interessantes. Nelas, a coordenada x de cada ponto é determinada pela função seno, e a coordenada y pela função co-seno:

```
AP LJ :COEF1 :COEF2 :PASSO
  DESENHE SEMCOR SEMT
  POS :COEF1 :COEF2 0 COLORIDO
  LJ1 :COEF1 :COEF2 0 :PASSO
FIM

AP POS :COEF1 :COEF2 :ANG
  FACA "X100 * SEN (:COEF1 * :ANG)
  FACA "Y100 * COS (:COEF2 * :ANG)
  DEFXY :X :Y
FIM

AP LJ1 :COEF1 :COEF2 :ANG :PASSO
  POS :COEF1 :COEF2 :ANG
  LJ1 :COEF1 :COEF2 (:ANG + :PASSO) :PASSO
FIM
```

Por tudo isso, você já deve ter se convencido do extraordinário potencial do LOGO para o tratamento de números. Por meio da construção de novos comandos a partir dos primitivos numéricos, podemos solucionar praticamente qualquer problema matemático com um mínimo de esforço. Outra vantagem apresentada pelo LOGO é que você pode visualizar o resultado, graças à tartaruga, além do resultado numérico normal que se obtém com o emprego das outras linguagens.

Exercício 8

A) Escreva uma rotina para calcular a enésima potência de um número, de modo que POTENCIA 42 tenha como resultado 16.

B) Elabore um conjunto de rotinas para converter em hexadecimal um número decimal (empregue a mesma técnica usada no exemplo dos números binários, mas desta vez divida por 16).

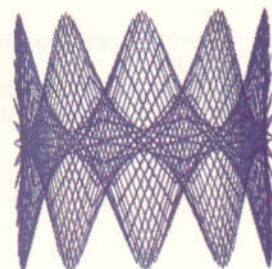
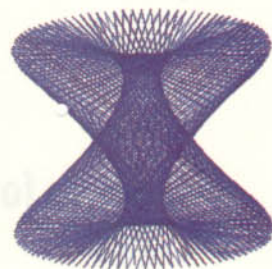
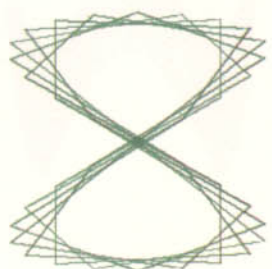
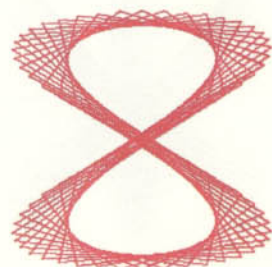
C) Escreva uma rotina PAR? que tenha como resultado VERD se o número for par, e FALSO se for ímpar.

D) Use o método Monte Carlo para determinar a área sob a curva $y = x^2$ entre $x = 0$ e $x = 10$.

Registre

Linha Apple

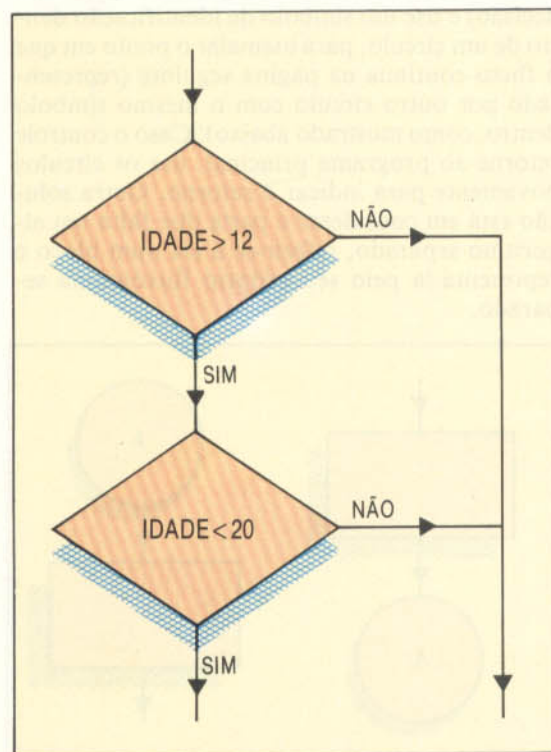
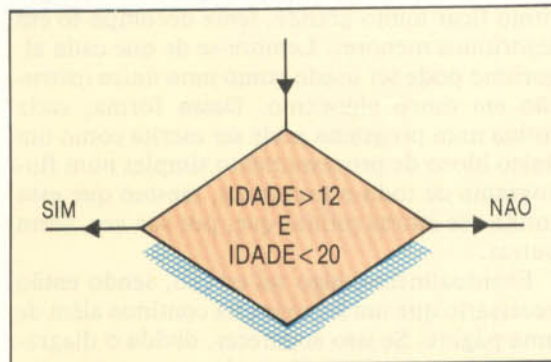
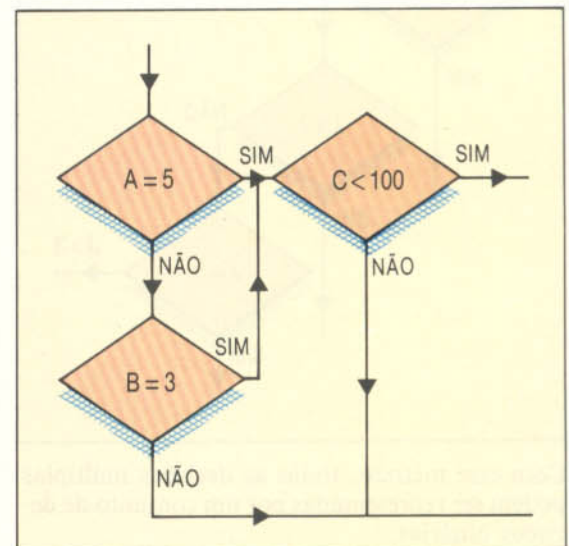
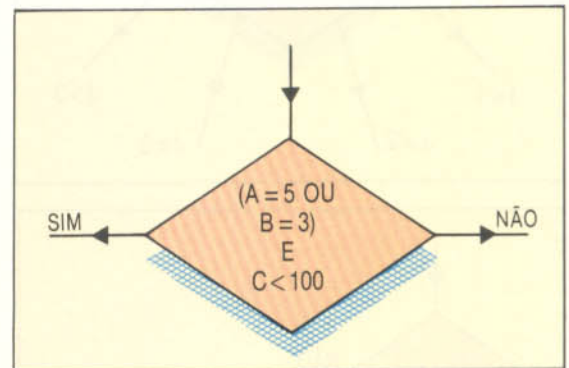
Este programinha em BASIC também produz um surpreendente efeito gráfico na tela de seu micro: uma série de linhas começa a varrer a tela, partindo do canto inferior esquerdo, e em seguida outro feixe se irradia a partir do canto direito. Há sete cores de linhas que se sucedem aleatoriamente. Veja as linhas 30 e 40 do programa: alterando a linha 30 e colocando-a em diferentes lugares do programa, você obterá belos padrões coloridos.

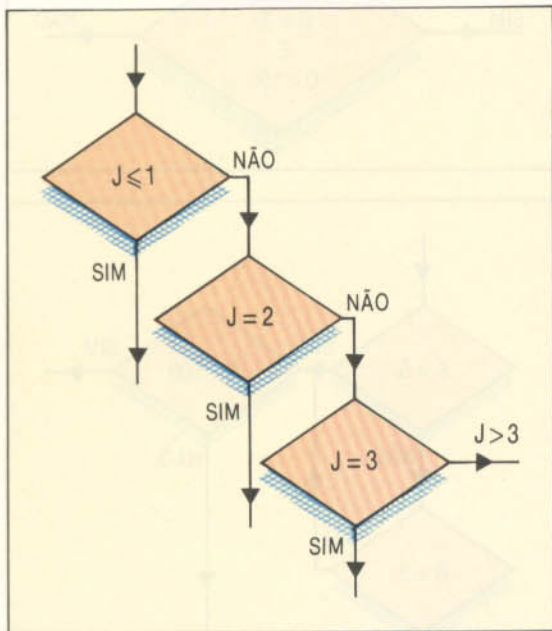
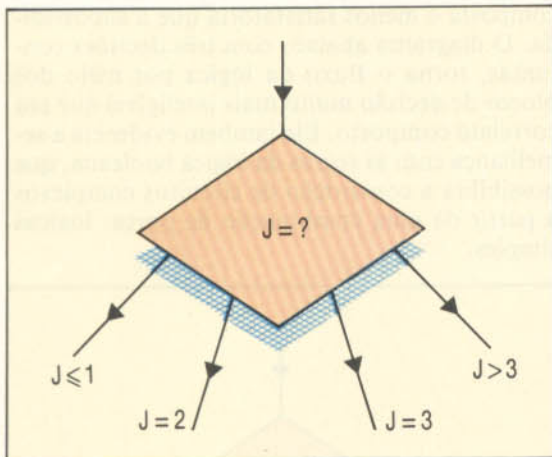
```
5 REM **** FEIXE DE LINHAS ***
10 HGR : HCOLOR = 3 : HOME
20 FOR N = 0 TO 159
30 W = INT (RND (1) * 7)
40 HCOLOR = W
50 HPLOT 0,N TO 159,N
60 HPLOT 0,N TO 159,159 - N
70 HPLOT N,0 TO N,159
80 HPLOT N,0 TO N,159 - N
90 NEXT N
100 FOR I = 1 TO 3000 : NEXT I
```

DECISÕES CRUCIAIS

Na utilização de fluxogramas para o desenvolvimento de programas, as decisões compostas podem dividir-se em componentes simples. Nos casos mais complexos, usam-se as tabelas de decisão.

Os programadores precisam usar às vezes decisões compostas em seus programas, como:

SE IDADE>12 E IDADE<20 ENTAO FAIXA = "ADOLESCENTE"

Os algoritmos com instruções como essa serão mais fáceis de entender se a decisão composta for subdividida nas decisões que a compõem.

Apresentamos o exemplo em BASIC sob a forma de um diagrama, para mostrar como a versão composta é menos satisfatória que a subdividida. O diagrama abaixo, com três decisões conjuntas, torna o fluxo da lógica por meio dos blocos de decisão muito mais inteligível que seu correlato composto. Ele também evidencia a semelhança com as regras da lógica booleana, que possibilita a construção de circuitos complexos a partir de uma combinação de portas lógicas simples.

Nos exemplos apresentados, todas as decisões são binárias, mas é bastante comum um algoritmo envolver decisões com mais de dois resultados possíveis. Se o programa aceitar uma entrada pelo teclado que signifique uma escolha num menu, será necessário desviar para uma sub-rotina. A maioria das linguagens de programação fornece estruturas de desvio como ON-GOTO e ON-GOSUB, em BASIC, e CASE-OF, em PASCAL. As regras para as decisões binárias também se aplicam às decisões múltiplas: pode-se escolher somente uma rota a partir de uma decisão, e todas

as saídas devem ser bem definidas, com as rotas possíveis sendo mutuamente exclusivas e abrangendo todas as possibilidades. É possível representar uma decisão múltipla, como no exemplo mostrado, com um conjunto de rotas de saída provindo da mesma decisão. No entanto, isso não é muito comum e, mais freqüentemente, a decisão terá de se decompor em várias decisões binárias, como no exemplo.

Com esse método, todas as decisões múltiplas podem ser representadas por um conjunto de decisões binárias.

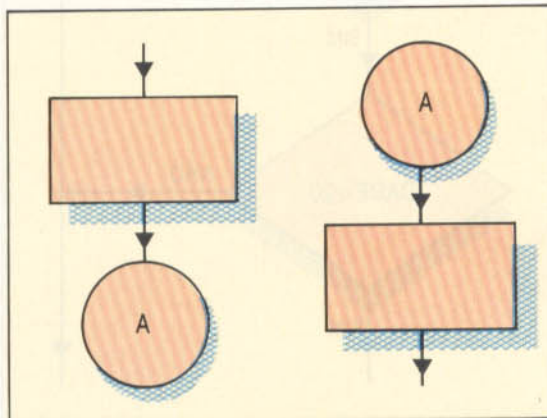
A tabela de decisão

Como alternativa aos fluxogramas, especialmente quando são possíveis muitas decisões múltiplas, recomenda-se usar tabelas de decisão. Esse tipo de tabela apresenta quatro partes principais: as condições para as regras; as ações a ser realizadas; um quadriculado, que mostra como as condições se encaixam nas regras; e mais um quadriculado, com as ações apropriadas para cada regra. No diagrama "condições/regras", os valores das variáveis aparecem nas células; já no diagrama "ações/regras", abaixo do primeiro,

	REGRAS							
CONDIÇÕES	1	2	3	4	5	6	7	8
BOTÃO DETONADOR ACIONADO	✓	✗	✗	✗	✓	✗	✓	✗
NÍVEL DO JOGO	1	1	2	2	1	1	2	2
NÍVEL DO JOGADOR	NOVATO	NOVATO	NOVATO	NOVATO	AVANÇADO	AVANÇADO	AVANÇADO	AVANÇADO
AÇÕES								
RETIRADA DOS ALIENÍGENAS							✓	✓
DETONAÇÃO ALEATÓRIA			✓	✓			✓	✓
REDUZIR O NÍVEL DE ENERGIA EM	1%	1%	2%	2%	2%	2%	4%	4%

uma marca indica que ação deve ser executada, e um valor age como seu parâmetro de entrada. A regra 4, por exemplo, é a seguinte: "Se o botão de disparo for pressionado, o nível do jogo for 2 e o jogador for novato, acione os detonadores aleatoriamente e reduza o nível de energia em 2%".

Sempre que possível, limite os fluxogramas a uma página. Pode ser irritante e demorado folhear várias páginas de papel. Quando seu algoritmo ficar muito grande, tente decompô-lo em algoritmos menores. Lembre-se de que cada algoritmo pode ser usado como uma única instrução em outro algoritmo. Dessa forma, cada rotina num programa pode ser escrita como um único bloco de processamento simples num fluxograma de todo o programa, mesmo que essa rotina use outras rotinas que, por sua vez, usem outras.

Eventualmente, algo sai errado, sendo então necessário que um fluxograma continue além de uma página. Se isso acontecer, divida o diagrama num ponto conveniente (por exemplo, uma decisão) e use um símbolo de identificação dentro de um círculo, para assinalar o ponto em que o fluxo continua na página seguinte (representado por outro círculo com o mesmo símbolo dentro, como mostrado abaixo). Caso o controle retorne ao programa principal, use os círculos novamente para indicar o retorno. Outra solução está em considerar a parte que falta um algoritmo separado, referir-se a ela num bloco e representá-la pelo seu próprio fluxograma separado.

O INÍCIO DO JOGO

Space Invaders
O mais conhecido dos jogos eletrônicos é um dos descendentes do Pong, criado por Nolan Bushnell.

Quando conectou um microprocessador a seu aparelho de televisão e inventou o jogo Pong (que, como o nome sugere, se assemelhava ao pingue-pongue), Bushnell deu início à história da Atari.

O método de Nolan Bushnell para colocar sob o controle do usuário as imagens que aparecem na tela mudou o conceito popular de lazer e conquistou milhões de jovens.

Ele e dois parceiros (Ted Dabney e Larry Bryan) investiram algum dinheiro e lançaram o jogo eletrônico Pong em Sunnyvale, Califórnia, em 1972. Logo perceberam que tinham em mãos um sucesso e uma fonte de lucros.

O domínio da Atari sobre o mercado de videogames começou com uma decisão oportuna tomada por essa empresa logo depois: comprar os direitos da invenção de Bushnell.

A Atari manteve a dianteira por quase toda a década de 70, até o gosto do público se transferir dos jogos do tipo fliperama para os micros. O mercado de jogos é como o de discos: devem-se descobrir as estrelas em potencial e promovê-las. Talvez por isso a Atari tenha sido comprada pela Warner Communications International, ligada a filmes e discos.

Console para jogos
O Sistema de Vídeo Computador (VCS, Video Computer System) da Atari é fornecido com dois joysticks, adaptador de cabos e um cartucho. Destinado exclusivamente a jogos, o VCS não pode ser usado como computador para aplicações gerais. Todos os jogos vêm em cartuchos.

Os fliperamas operados por moedas ou fichas proporcionaram lucros enormes até os últimos anos da década de 70. Mas logo o negócio entrou em declínio e, em meados da década seguinte, causou prejuízos consideráveis.

Space Invaders, da Taito, habilmente comercializado pela Atari, é o mais conhecido de todos os jogos para computador. Tornou-se um fenômeno social e originou uma enxurrada de outros cartuchos com temas galácticos. A empresa esteve no centro da febre dos jogos eletrônicos, com vários de seus produtos nas listas dos preferidos: Asteroids, Battlezone, Centipede, Missile Comand e The Tempest, por exemplo.

A obsessão pelos fliperamas foi meteórica. Os freqüentadores logo se voltaram para os microcomputadores, que ofereciam duas grandes vantagens: o mesmo divertimento das casas de jogos eletrônicos, sem dispêndio de dinheiro. Além disso, o usuário passava a possuir uma máquina de computação bastante flexível.

Inicialmente, a Atari respondeu a essa mudança na preferência dos consumidores convertendo seus melhores softwares para fliperama em jogos destinados a micros. Conectava-se um cartucho ao micro, expandindo ou substituindo a ROM do próprio computador. Assim, não era necessário carregar o jogo na memória a partir de um cassete ou disquete. A tecnologia necessária, porém, tornava os cartuchos muito caros e não reprogramáveis. Em conseqüência, a empresa se via freqüentemente com pilhas de encalhes dos jogos impopulares.

O grande declínio
Durante a maior parte da década de 70, a Atari prosperou com os lucros gerados por máquinas de fliperama como a Major Havoc. O advento do microcomputador, contudo, exigiu uma nova estratégia de mercado.

Declínio da fortuna

A estratégia de mercado utilizada pela Atari para superar o problema não deu bons resultados. Baseando as projeções de venda num jogo que alcançara enorme sucesso — o PacMan —, ela pagou um alto preço por esse otimismo exagerado: catorze caminhões despejaram os cartuchos não vendáveis num grande buraco no deserto de Nevada.

A Atari também falhou por não investir numa característica operacional fundamental dos computadores: as instruções de um programa não precisam converter-se em entidade física para serem distribuídas. Sua transmissão pode ser feita por telefone, cabo, rádio ou televisão. Novos produtos e técnicas que permitem essas formas de transmissão tornam-se cada vez mais disponíveis no mercado internacional.

Os problemas da Atari foram causados, em parte, pelo desentendimento entre a divisão de videogames, que estava em declínio, e a crescente divisão de microcomputadores. Agora, as duas se integraram.

Os microcomputadores da empresa têm como característica três chips especiais (Pokey, Antic e GTIA), que controlam, respectivamente, as portas de entrada e saída, os gráficos e a cor. Todos utilizam o processador 6502 e há grande variedade de programas à disposição.

FATOS NAS PONTAS DOS DEDOS

O gerenciador de banco de dados é um dos tipos mais importantes de software para armazenamento de dados. Aqui, você vai saber em detalhes seu uso nos microcomputadores, suas limitações e como tirar vantagem dele.

Pode-se definir um banco de dados como uma coleção de informações sobre determinado assunto. Um banco de dados guarda conjuntos de informações como estas: nomes e endereços dos sócios de um clube; despesas decorrentes das reuniões do clube; datas e locais dessas reuniões; ou o pagamento das taxas de manutenção pelos sócios. Contudo, um sistema de banco de dados poderia incluir facilmente todos esses conjuntos de informações num grande arquivo.

Um arquivo é uma coleção de registros, que por sua vez abrangem diversos campos. Você pode imaginar um registro como uma ficha contendo informações sobre uma pessoa, em que os espaços para nome e endereços, por exemplo, são campos. Em alguns programas aplicativos, o software fixa a estrutura do arquivo, mas num sistema de banco de dados é mais comum escolher-se a estrutura de acordo com as tarefas. Isso implica a determinação do tamanho de cada campo e a definição do tipo de conteúdo a ser armazenado nele. Num arquivo de nomes e endereços, por exemplo, as informações sobre cada pessoa ocupam um registro: o nome é armazenado num campo de caracteres; e cada linha do endereço, em campos separados. Esses campos têm geralmente trinta caracteres.

Além dos campos de caracteres, existem também campos numéricos; o programa faz cálculos com os dados neles armazenados e dessa maneira produz informações úteis. No hipotético arquivo de sócios do clube, o programa poderia calcular quantos sócios haviam pago as taxas, qual a receita anual acumulada, ou quanto falta ser recebido. Contudo, nem todos os dados numéricos devem ser armazenados num campo numérico. Números de telefones são um bom exemplo de dados com os quais jamais se fará qualquer cálculo.

Antes de saber se um arquivo pode ser usado com eficiência em seu computador, você precisa calcular o tamanho máximo (número de registros) que ele terá e a quantidade de memória necessária para cada registro, e então determinar se o equipamento tem memória suficiente para armazená-lo.

Coleção de informações
A maioria das pessoas guarda suas informações de modo desordenado, em pedaços de papel. Para elas, um banco de dados feito em microcomputador é ideal. Pode conter registros sobre o seguro do carro ou da casa, números de contas bancárias ou de telefones, e tudo o mais que seja classificável numa ordem determinada, como os dados de uma coleção de selos, moedas, minerais etc.

Ligações perigosas
No passado, os bancos de dados ficavam distanciados entre si tanto pelo espaço quanto pelo tempo. Hoje, com a possibilidade da troca de informações, chegam a ameaçar a privacidade das pessoas: bancos de dados são cada vez mais numerosos nas sociedades tecnológicas e o ritmo de acesso a eles se intensifica, porque as sociedades dependem mais e mais de informações. No quadro abaixo, algumas fases da evolução dos registros de dados.

Um registro com um nome e três campos de endereço de trinta caracteres cada, mais um número de telefone de dez caracteres, ocupa um total de 130 bytes. Se você tem um sistema de disco que armazena, digamos, 200 bytes por disco, então pode guardar até 1.500 registros em cada disquete. Num sistema com 48 bytes de memória, mas que use fita magnética, provavelmente só será possível manter cerca de trezentos registros (ocupando-se 10 bytes com o programa e o sistema operacional).

Na prática, se você deseja colocar esse arquivo em ordem ou fazer outras manipulações, precisará de memória extra; assim, convém limitar o tamanho do arquivo em fita até a metade da área disponível para ele.

Os dois sistemas de armazenamento — em disco e em fita — diferem tanto porque os arquivos em fita devem ser inteiramente passados para a memória do micro e processados em conjunto, ao passo que a velocidade e o modo de acesso aos discos permitem que os arquivos gravados neles sejam processados por etapas.

Registros, da Idade Média à atualidade

Familiares
Nobiliárquicos
Fundiários
Contábeis
Eclesiásticos
Jurídicos
Das cidades medievais
Financeiros
Paroquiais
Municipais
Bancários e comerciais
Civis
Da Receita Federal
Policiais
Creditícios
Assistenciais
Gerais

Um sistema de gerenciamento de banco de dados possibilita tirar informações de um arquivo — por exemplo, o total das despesas do ano. Ligue-o com uma informação de outro arquivo, como a receita total das taxas pagas pelos sócios, ou faça comparações entre as duas. Você pode, então, com base nessas informações, decidir o que deve ser feito: abrir a venda de títulos novos, diminuir o número de reuniões ou gastar os lucros na compra de um computador para o clube.

Ou pode querer enviar uma carta a todos os sócios, informando-os sobre a situação. Nesse caso, o banco de dados forneceria os nomes e endereços e os imprimiria diretamente nos envelopes ou em etiquetas adesivas. Ou, então, ele poderia ser ligado a um programa de processamento de texto, para escrever cartas ou informes, fazendo automaticamente o endereçamento.

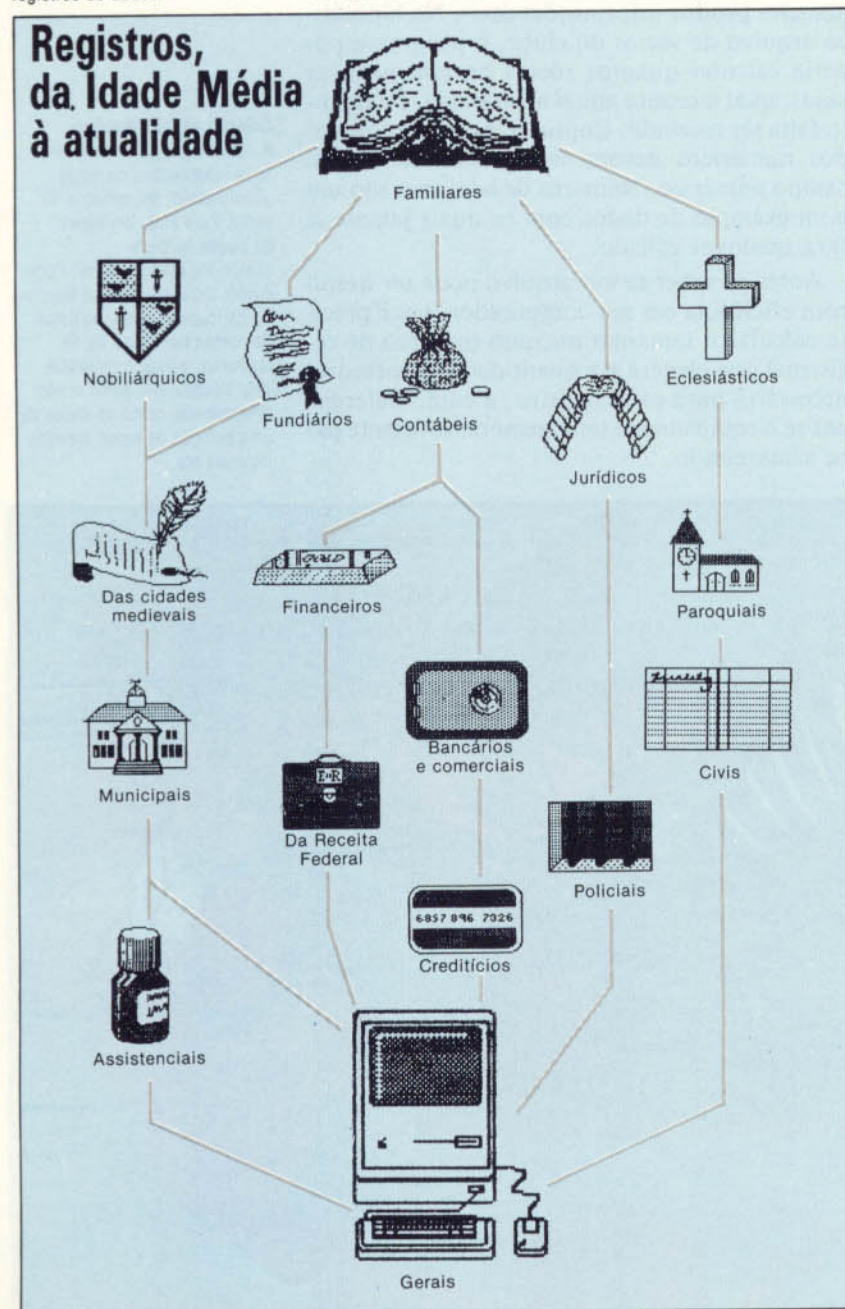
Embora muitas coisas possam ser feitas por um gerenciador de banco de dados ideal, não é fácil encontrar um que venha com todas as facilidades juntas, sobretudo se você tiver um computador de uso doméstico com espaço restrito de memória e armazenamento em fita ao invés de em disco.

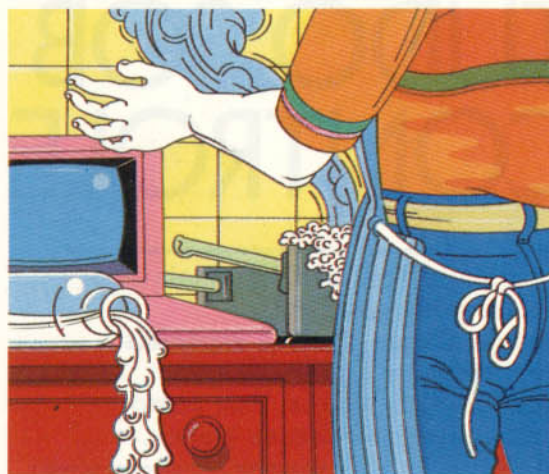
Muitos dos pacotes de bancos de dados menos sofisticados trabalham com apenas um campo por vez, de maneira que, mesmo podendo guardar diferentes conjuntos de informações, você não poderá ligá-los ou compará-los no verdadeiro estilo do banco de dados.

Um perigo real em todo processamento de dados é colocar sua preciosa informação num computador e então, por acidente, apagá-la. Ao lidar com informações importantes, é fundamental manter cópias de segurança e imprimir os dados sempre que houver atualizações, para que nada possa ficar irremediavelmente perdido.

Um pacote simples de banco de dados processa um arquivo por vez. O sistema deve fazer o trabalho de rotina de colocar a informação no arquivo tão facilmente quanto se preenche um formulário, e deve permitir que você tenha acesso à informação tanto na tela quanto por meio da impressora. O usuário seleciona registros por meio de critérios de busca e seleção, como, por exemplo, "todos os sócios que não pagaram este ano e têm o CEP xis".

Um pacote deste tipo também deve permitir a impressão de campos como os de nomes e endereços, para a emissão de etiquetas de endereçamento. Esse tipo mais simples de pacote existe para a maioria dos computadores de uso doméstico que funcionam com fita. No entanto, se você tem um sistema de disco, a variedade de software disponível é muito maior.

Pacotes mais sofisticados permitem que você desenhe um formulário de entrada de dados na tela para combinar com um sistema anterior ba-

Receita que desanda

Heitor Megabyte tem um pequeno micro, gosta muito dele e de mostrar tudo o que sabe. A máquina ajudou-o a entender os computadores e a aprender um pouco de programação em linguagem BASIC, mas, infelizmente, Heitor ainda não sabe a diferença que existe entre aplicação séria e diversão. Ele insiste em usar o computador para guardar os números dos telefones de seus (poucos) amigos e tem um software para banco de dados em fita de que se vale para guardar receitas culinárias. Quando Heitor convida alguém para jantar, a comida, invariavelmente, só sai por volta da meia-noite, porque ele insiste em conferir os detalhes de cada receita do micro. Para isso, tem de ligar o computador na tomada com gestos teatrais, carregar o programa em fita (o que em geral exige várias tentativas), esperar durante intermináveis minutos enquanto o micro procura a receita, ler as instruções na tela, aos poucos, e finalmente preparar o jantar. Não adianta chegar mais tarde, porque Heitor não começa a preparar a comida sem a exibição. Para quem quer ver um micro gerando eficiência, é uma lástima.

Escolha sua música

Faz tempo que a coleção de discos de Clara Chip é grande. E ela resolveu aproveitar esse fato para alugá-los. A experiência ensinou-lhe que é fundamental oferecer os discos certos a cada tipo de ouvinte, e por isso ela decidiu comprar um computador. Para ter velocidade e alta capacidade de armazenamento, escolheu um sistema com dois drives. Então, escreveu num papel os nomes das categorias de música que gostaria de incluir em seu software de banco de dados. Havia dois grupos principais (música erudita e música popular), cada um com subtítulos, que eram também subdivididos. Clara usa o gerenciador de banco de dados para escolher os discos de acordo com o cliente — digitando, por exemplo, o comando para listar todos os discos de samba e então fazendo uma seleção mais detalhada, em nível de bossa nova ou carnaval. Ela examina a listagem no monitor e em seguida tira uma cópia dela na impressora. Depois, é só ir à estante e apanhar o disco escolhido para o cliente.

seado em papel. Outros pacotes possibilitam selecionar itens numéricos (por exemplo: maiores ou menores que determinada quantia). Portanto, é importante procurar um pacote adequado a suas necessidades específicas.

Arquivos interligados

Um computador com gerenciador de banco de dados é útil para tarefas que envolvem repetição, ou nos casos em que uma grande quantidade de informação precisa ser armazenada ou examinada rapidamente. Mas há algumas aplicações que não são adequadas para os sistemas de banco de dados: não seria sensato vasculhar a lista telefônica de um computador doméstico toda vez que você quisesse fazer uma ligação. No tempo gasto para ligar o computador, carregar o programa de banco de dados, chamar o arquivo e procurar o número, você poderia ter feito seis ligações, pegando os números de uma agenda.

Um pacote de banco de dados mais potente proporciona todas essas facilidades e você ainda pode usar a matemática para obter dados tais como a renda total proveniente das taxas de manutenção do clube, ou calcular qual a instalação mais usada nele.

Vários arquivos de informações podem ser interligados, de modo que os dados sobre assuntos correlatos sejam usados em conjunto. Por exemplo, determinado jogador pertencente a nosso clube imaginário tem seus dados pessoais registrados num arquivo e os detalhes de seu desempenho nos torneios do clube em outro. Quando for pedido um histórico sobre ele, os dois arquivos poderão ser usados em conjunto. Para se ter uma posição do desempenho de todos os jogadores do clube em determinada data, consulta-se o arquivo chamando-se todos os registros daquela data em vez de todos os registros de cada pessoa.

Para executar um programa com facilidades tão extensas, seria necessário usar um computador de mesa ou profissional, com um mínimo de 64 Kbytes de memória e sistema de disco.

Os gerenciadores de bancos de dados têm condição de processar qualquer tipo de informação, desde simples listas até sistemas altamente complexos, com arquivos múltiplos, e têm facilidade para criar relatórios elaborados e capacidade de processar informação de um modo que pode ser transferido para outro software, com os processadores de textos.

TUDO SOB CONTROLE

ELITE
ABCDEFGHIJKLMNOP
QRSTUVWXYZabcdef
ghijklmnopqrstuv
wxyz0123456789 !
"£$%&'()*+,-./:;
<=>?@[\]^_'{|}~

PICA
ABCDEFGHIJKLM
NOPQRSTUVWXYZ
abcdefghijklm
nopqrstuvwxyz
0123456789 !"
£$%&'()*+,-./
:;<=>?@[\]^_'
{|}~

EMPHASISED PICA ITALIC
ABCDEFGHIJKLM
NOPQRSTUVWXYZ
abcdefghijklm
nopqrstuvwxyz
0123456789 !"
#$%&'()+,-./*
:;<=>?@[¥]^_'
{|}~

ENLARGED ELITE
A B C D E F G H
I J K L M N O P
Q R S T U V W X
Y Z a b c d e f
g h i j k l m n
o p q r s t u v
w x y z 0 1 2 3
4 5 6 7 8 9   !
" £ $ % & ' ( )
* + , - . / : ;
< = > ? @ [\ ]
^ _ ' { | } ~

DOUBLE-STRIKE CONDENSED PICA
ABCDEFGHIJKLMNOPQRSTUVW
XYZabcdefghijklmnopqrst
uvwxyz0123456789 !"£$%&
'()*+,-./:;<=>?@[\]^_'{
|}~

Pontilhados
A impressora matricial possui uma variedade de tipologias (Pica, Elite e Italic) e modelos de tipos (condensados, ampliados e com dupla impressão). Os exemplos acima foram impressos na Epson FX-80.

Até a impressora matricial mais simples pode gerar uma série de efeitos gráficos especiais que melhoram a apresentação visual de documentos. Veja como conseguir alguns desses efeitos.

Impressoras do tipo matricial fazem muito mais do que produzir listagens de programas. Uma variedade de efeitos especiais também pode ser executada por elas.

Até a impressora mais simples permite que se altere o tamanho dos caracteres impressos. Em geral, um texto é impresso com oitenta caracteres por linha, número que pode ser aumentado, selecionando-se o modo "impressão condensada" (que usa caracteres menores), ou diminuído, escolhendo-se a "impressão ampliada". O espaço entre as linhas se altera da mesma forma.

A impressora que examinaremos em detalhes é um bom exemplo de máquina com grande variedade de tipos de impressão. Você pode optar pela dupla impressão da letra (negrito), que imprime o texto com caracteres mais escuros, ou pela impressão em itálico. Uma de suas características mais interessantes é a capacidade de mudar qualquer dos caracteres armazenados na sua memória, o que facilita, por exemplo, a impressão de símbolos científicos. Para começar, vamos ver como uma impressora realiza a simples tarefa de imprimir a listagem de um programa.

A maneira pela qual um computador "fala" com uma impressora varia conforme a máquina. O Dragon, por exemplo, usa uma variação simples do comando LIST — o LLIST —, instruindo a impressora para produzir uma cópia do programa. Outras máquinas precisam da abertura de "canais" ou "correntes" para ter acesso à impressora. Para saber como realizar essa operação, o mais indicado é consultar o manual do computador.

Estabelecida a comunicação entre as duas máquinas, prepare-se para enfrentar alguns problemas. Os mais freqüentes são o do texto ilegível ou o das linhas em branco entre as linhas do programa. A explicação para esses defeitos está na diferença entre um caractere "avanço de linha" (Line Feed ou LF) e um caractere "retorno do carro" (Carriage Return ou CR). Depois que envia uma linha do texto para a impressora, o computador também envia um caractere de retorno do carro que faz a cabeça de impressão voltar para a margem esquerda, pronta para imprimir nova linha. Alguns computadores também enviam um caractere de avanço de linha para mover o papel para a linha seguinte; outros, não, presumindo que a impressora faça isso automaticamente. A maioria das impressoras possui um interruptor interno que define se a impressora gera seus próprios sinais de avanço de linha. Se ocorrerem alguns desses problemas, mude o interruptor para a posição alternativa.

Além de produzir listagens de programas, uma impressora pode ser usada como dispositivo de saída — em vez de apresentados na tela, os caracteres são impressos no papel. A operação desse processo também varia de um computador para outro. Os exemplos de programação aqui apresentados usam LPRINT, e talvez você tenha de fazer uma alteração para adaptá-los a sua máquina.

Etiquetas de endereçamento

```
10  LPRINT "SR. L. HENRIQUE ALAYON"
20  LPRINT "AV. FARIA LIMA, 1106/709"
30  LPRINT "01452  SAO PAULO     SP"
40  FOR I = 1 TO 8
50  LPRINT
60  NEXT I
70  GOTO 10
```

Esta listagem é um program~ simples para produzir etiquetas de endereçamento. Como não usa código especial de controle, o programa funcionará com qualquer marca de impressora. Como está, o programa imprimirá o mesmo nome e endereço repetidamente.

Talvez você queira alterar isso, para colocar nomes e endereços diferentes, ou mesmo para que eles sejam lidos de um arquivo de dados. O loop FOR-NEXT entre as linhas 50 e 70 imprime sete linhas em branco e é usado para posicionar a cabeça da impressora corretamente no começo de cada etiqueta. O número exato de linhas em branco precisa ser ajustado para sua máquina.

Nosso programa é adequado para etiquetas de impressão simples, mas, para imprimir algo mais complexo, como uma fatura ou um cabeçalho, será preciso usar alguns efeitos especiais. Eles são produzidos enviando-se códigos de controle para a impressora e também para os caracteres normais do texto.

Além de um código para cada caractere no teclado, o Padrão ASCII tem um grupo de caracteres "invisíveis", que não imprimem nada na tela ou no papel. São estes os códigos usados para produzir os efeitos especiais da impressora: no conjunto ASCII padrão há quatro códigos

(17, 18, 19 e 20) que são reservados para comandos de controle de dispositivos. O conjunto de caracteres ASCII não possui reserva suficiente de comandos de controle para as setenta e tantas funções de uma Epson FX-80. A fim de resolver esse problema, a maioria dos efeitos é produzida enviando-se "códigos de escape" para a impressora. Eles consistem em dois ou mais códigos de caracteres, começando com um caractere ESC (código ASCII 27). Por exemplo, para ligar a função de espaçamento proporcional numa Epson, você envia ESC-p — isto é, o caractere Escape seguido do caractere "p" minúsculo.

Em BASIC, isso é escrito assim:

```
LPRINT CHR$(27);"p"
```

O caractere ESC não pode ser produzido normalmente apertando-se a tecla [Escape]. Por isso usa-se a função CHR$.

No BBC, você usaria:

```
VDU2
VDU1, 27,1, 112
VDU3
```

O comando VDU2 liga ("ativa") a impressora; VDU1 significa "envie o seguinte caractere somente para a impressora". PRINT também mandaria o caractere ESC para a tela, mas sem o mesmo resultado. VDU3 desliga ("desativa") a impressora.

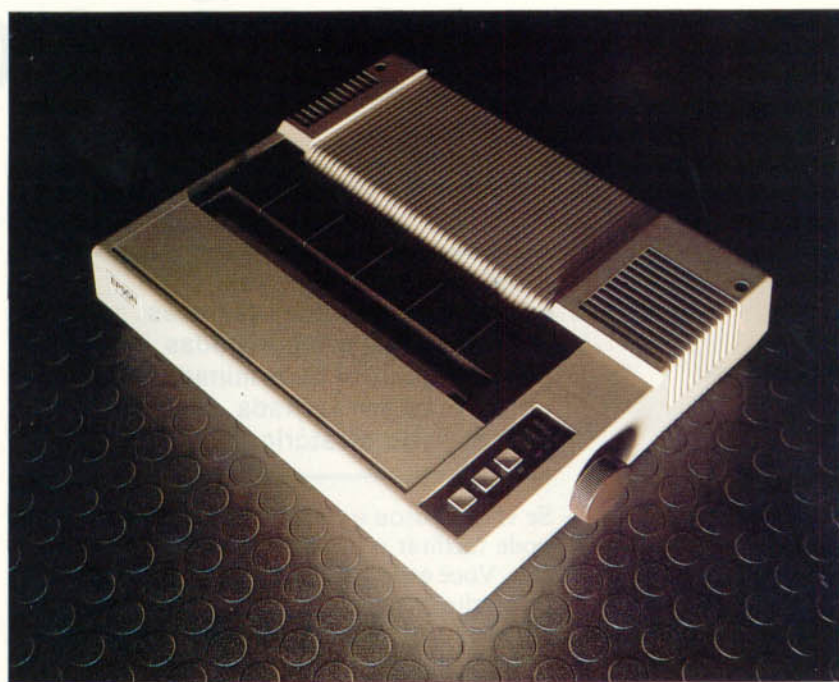
Essas seqüências de comando aplicam-se unicamente à Epson FX-80. Se você tentar a mesma seqüência de código para uma impressora diferente, ela não produzirá efeito, poderá acontecer algo inesperado ou até desligar a impressora (isto é, ela se recusará a responder ao computador).

Criando uma fatura

Nossa segunda listagem demonstra o uso de algumas das funções da Epson para criar o cabeçalho de uma fatura, usada por uma pequena oficina. Os códigos aqui aplicados são os da Epson FX-80. Se sua impressora for incompatível com a Epson, você deve adequar os códigos de controle.

```
999  REM *** CABECALHO DE FATURA ***
1000 LPRINT CHR$(12)
1010 LPRINT CHR$(14);TAB(12);"ABC
     MOTORES LTDA."
1020 LPRINT CHR$(13);CHR$(13)
1030 LPRINT CHR$(27);"E";
1040 LPRINT TAB (36);
1050 LPRINT CHR$(27);"-";CHR$(1);
1060 LPRINT "FATURA";
1070 LPRINT CHR$(27);"-";CHR$(0);
1080 LPRINT CHR$(27);"F";
1090 LPRINT CHR$(13);CHR$(13);CHR$(13)
```

Para começar, a linha 1000 envia o caractere com o código 12 para a impressora. Este é o caractere de "avanço de página" (FF), que dá instruções à impressora para correr o papel até o começo de uma nova folha. Depois, temos o código ASCII 14: ele é chamado de caractere de "shift out" (SO) e, na Epson, faz com que todo

Epson FX-80
É uma impressora popular entre os usuários de micros comerciais e de emprego doméstico. Tem uma cabeça de nove agulhas e sua velocidade máxima de impressão chega a 160 caracteres por segundo. No Brasil, uma impressora similar à Epson é a Grafix.

o texto subseqüente seja impresso em letras ampliadas. Em nosso programa, é usado para o cabeçalho, escrevendo o nome da oficina em letras grandes. A função TAB é usada para centralizar o cabeçalho.

CHR$(13) é o caractere de retorno do carro (CR), que produz uma linha em branco quando impresso por si só. É usado várias vezes entre as linhas 1020 e 1090, para espaçar o cabeçalho da fatura. ESC-E, na linha 1030, liga o modo de dupla impressão: com isso, o texto que vem a seguir é impresso em tipos mais escuros. A linha 1050 liga a função de "sublinhar" e a linha 1070 desliga-a, depois de imprimir e sublinhar a palavra "FATURA". ESC-F anula o modo de dupla impressão. O texto impresso ficará assim:

ABC MOTORES LTDA.

FATURA

Essa é apenas a parte inicial. Um programa completo de fatura incluiria linhas para imprimir detalhes do cliente, como nome, marca do carro, valor a pagar etc. Esses detalhes seriam obtidos a partir de uma série de perguntas feitas no início do programa e as respostas ficariam armazenadas como variáveis.

Os dois programas aqui apresentados são exemplos simples dos usos alternativos de uma impressora do tipo matricial. Assim, programar sua impressora pode ser tão divertido quanto programar o próprio computador.

POSIÇÕES-CHAVES

A indexação de registros e o hashing são duas técnicas de acesso muito empregadas para se localizar determinada informação armazenada num registro aleatório.

Se você já usou o acesso aleatório, sabe que ele pode facilitar o uso de arquivos em programação. Você especifica qualquer registro a ser lido ou escrito, sem ter de preocupar-se com os complicados procedimentos necessários para recuperar informações armazenadas seqüencialmente.

Contudo, os métodos de inserção e eliminação não são os mais eficientes para usar arquivos. Um método bem melhor de acesso à informação é o índice.

Para criá-lo, determinado campo de cada registro deve ser especificado como *chave*. O conteúdo desse campo será usado para localizar o registro para consulta ou processamento. Assim, os valores de tais campos-chaves de cada registro, junto com o número desse mesmo registro, formam um índice. Desse modo, se o registro, digamos, de Leila Brito, tem o número 17 e é o oitavo numa listagem em ordem alfabética, então o algoritmo (a fórmula) do índice colocará o número 17 na oitava posição dessa listagem. Se o índice for periodicamente ordenado, a localização de um registro poderá ser um processo muito rápido.

Arquivos índices

Em geral, o índice fica na memória do computador durante o processamento, para que seu acesso seja imediato. Ele é gerado em instantes, por uma rotina que lê o arquivo inteiro, trazendo o conteúdo de cada campo-chave para uma matriz. Esse índice pode ser ordenado para uso imediato, mas isso implica grande demora.

Uma saída é armazenar vários índices como arquivos, no disco, junto com arquivos de dados. Dessa forma, cria-se qualquer número de campos-chaves para um registro, gerando para cada tipo de campo um arquivo índice guardado em disco. O arquivo pode ser indexado de várias maneiras: por exemplo, com os registros organizados em ordem alfabética (por nome), cronológica (por data) e assim por diante.

E como se armazena um arquivo índice? Para cada registro indexado ele precisa conter dois campos: a informação do campo-chave e o número do registro. Isso será transferido do disco para a memória, pronto para uso.

Esta é uma excelente aplicação para um arquivo seqüencial, no qual, ao contrário do que acontece no aleatório, a informação é solicitada na ordem em que foi armazenada. Nesse ponto, os arquivos de acesso seqüencial e os de acesso aleatório se complementam.

A remoção de registros inúteis nos arquivos aleatórios indexados é uma questão de marcá-los como anulados e assegurar-se de que eles não mais constem do índice. A maneira mais simples de fazer isso é colocar o sinal de "anulado" (um asterisco no começo do primeiro campo do registro, digamos). Com isso, a chave para esse registro pode ser removida do índice; ou, ainda, o número do registro, receber um valor especial, indicando anulação.

Qualquer que seja o método escolhido, porém, é importante que haja um controle dos registros anulados. Quando um novo for acrescentado ao arquivo, pode ficar sobreposto a um anulado. Nesse caso, a entrada do índice original no disco deve ser substituída por uma nova, e num ponto conveniente; e o índice deve ser reordenado, para incluir o novo registro na posição correta dentro do arquivo. Dessa forma, um programa permite a recuperação de registros anulados por acidente (desde que nada tenha sido escrito sobre eles).

Convém providenciar uma forma de agrupar os registros do arquivo de índices, porque o sistema que apresentamos permite que eles sejam armazenados fora de ordem, com numerosos e desnecessários intervalos entre um e outro. À medida que o arquivo cresce, a velocidade de acesso diminui. A rotina de agrupamento deve colocar os registros numa ordem conveniente e eliminar os anulados.

O hashing

A indexação não é a única maneira de localizar registros num arquivo com grande rapidez. Um método alternativo é o hashing, adequado para arquivos muito grandes e que, por essa razão, só é encontrado em sistemas de discos rígidos ou em máquinas com discos flexíveis de alta capacidade. Contudo, muitos sistemas operacionais e programas usam o hashing para acelerar as operações.

O hashing substitui o índice por um algoritmo. Ele toma o valor contido no campo-chave e a partir dele produz um número (o hash). O registro é então armazenado na posição do arquivo indicada por aquele número.

Planeja-se a fórmula de acordo com o tipo de informação do campo-chave. Se ele contém a data, por exemplo — permitindo que os registros

sejam organizados cronologicamente —, pode-se usar o número do mês, multiplicado pelos dois últimos dígitos do ano, mais o número do dia, para produzir o hash.

Suponha que desejamos criar um arquivo com o hash dos registros de empregados, usando sobrenomes como chave. O algoritmo que se pode usar é o seguinte: pegue os códigos ASCII das quatro primeiras letras e trate-os como um número de oito dígitos; eleve esse número ao quadrado e então use os quatro últimos dígitos do resultado como hash. JONES, portanto, é registrado como 1161, enquanto JONQUIL é registrado como 0161.

O hashing é muito diferente de um sistema de indexação. Com ele, você só pode ter um campo-chave (e um algoritmo) por arquivo; e ele é usado toda vez que se coloca um registro no arquivo.

O hashing mostra-se menos flexível do que a indexação, porém mais rápido. Para encontrar determinado registro, o programa apenas pega a chave, gera o hash e recupera o registro desejado.

Um problema ocorre quando dois registros geram o mesmo código e portanto deveriam ocupar a mesma posição num arquivo. Para evitar isso, os algoritmos são cuidadosamente planejados para que jamais duas chaves (a não ser as idênticas) gerem o mesmo hash. Além disso, os registros são espaçados no arquivo, de modo que dois hashes, aparentemente próximos um do outro, na realidade fiquem separados por uns cinco registros sem uso.

Quando se armazena um registro, sua chave é tratada para produzir um número hash de registro. Se o registro estiver ocupado, o sistema procura o registro seguinte pela ordem seqüencial. Isso pode ser feito com um bloco inteiro de uns cinco registros associados com aquele número hash.

Se um registro deve ser localizado, sua chave é tratada pelo algoritmo do hashing e aquele grupo de registros é então examinado seqüencialmente até encontrar-se a combinação perfeita. Isso parece anular a vantagem da velocidade, mas o hashing reduz o número de registros a serem examinados de aproximadamente 3 mil para apenas cinco ou seis.

O hashing também acelera a anulação de registros. Você faz o hash da chave do registro, dá uma busca rápida para localizá-lo e marca sua posição como vazia. Então, da próxima vez que um registro com idêntico hash for acrescentado ao arquivo, ele será escrito por cima do anulado.

E o que acontece quando os cinco registros para determinado hash estiverem ocupados? Há diversas maneiras de resolver isso. A mais óbvia consiste em enviar uma mensagem de arquivo cheio. Freqüentemente, os registros que não cabem são colocados num arquivo de excedentes, com seu próprio índice, e incorporados ao arquivo principal quando possível. Na maioria dos sistemas, há grande empenho em evitar excedentes, mantendo-se arquivos hash com menos de 80% de sua capacidade total.

Índice acoplado

Localize "Davi"

Um índice é o modo mais comum de orientar a busca de registros dentro de um arquivo aleatório: é uma lista de valores contidos em determinado campo de cada registro, armazenada na memória do computador enquanto o arquivo está sendo usado. Desse modo, pode ser examinada instantaneamente. Neste exemplo, registros inúteis ganharam o número −1, para serem depois substituídos por novos registros.

Chave	N.º
Aníbal	1
Beatriz	−1
Bonifácio	5
Cláudia	−1
Davi	7
Donato	23
Fábio	15
Gregório	28
Heitor	37
João	25
Klaus	11
Maíra	10

Arquivo índice

Arquivo principal

	Nome	Tel. com.	Tel. res.	Profissão
1	Aníbal	242 0791	727 0942	Desenhista
2	Pedro	636 2418	221 3940	Contador
3	Samuel	631 0836	286 8170	Editor
4	Registro cancelado			
5	Bonifácio	729 8213	236 2190	Fornecedor
6	Paulo	836 6622	298 4310	Decorador
	Davi	743 7216	450 6926	Jardineiro
8	Registro cancelado			
9	Registro cancelado			
10	Maíra	730 6321	429 7592	Estoquista
11	Klaus	493 9899	455 8431	Advogado
12	Wagner	736 7700	693 0452	Cabeleireiro

O hashing

Localize "Davi"

ALGORITMO DO HASHING

O hashing usa o conteúdo de uma chave para achar um grupo de registros, entre os quais está o procurado. Feito isso, a localização dele é instantânea.

O acréscimo de novos registros é feito nos espaços entre os blocos.

Nome	Tel. com.	Tel. res.	Profissão
Daniel	629 0491	430 0592	Funileiro
Dimas	436 2488	362 0066	Jornalista
Darci	730 0021	626 9191	Faxineira
Doroti	439 9933	630 4918	Psicóloga
Davi	743 7216	450 6926	Jardineiro
Dráusio	830 0123	340 9924	Enfermeiro
Ernesto	731 6666	458 0021	Desenhista
Eraldo	831 8294	450 6218	Dentista

Arquivos construídos com base em hashing oferecem alta velocidade de acesso aos registros, mas têm restrições que exigem muito cuidado na programação. Em troca, reduzem de 3 mil para cinco ou seis registros o volume de busca, quando se compara, digamos, com uma busca seqüencial. Para acrescentar, consultar ou eliminar registros, a velocidade de acesso ao arquivo é sempre muito elevada.

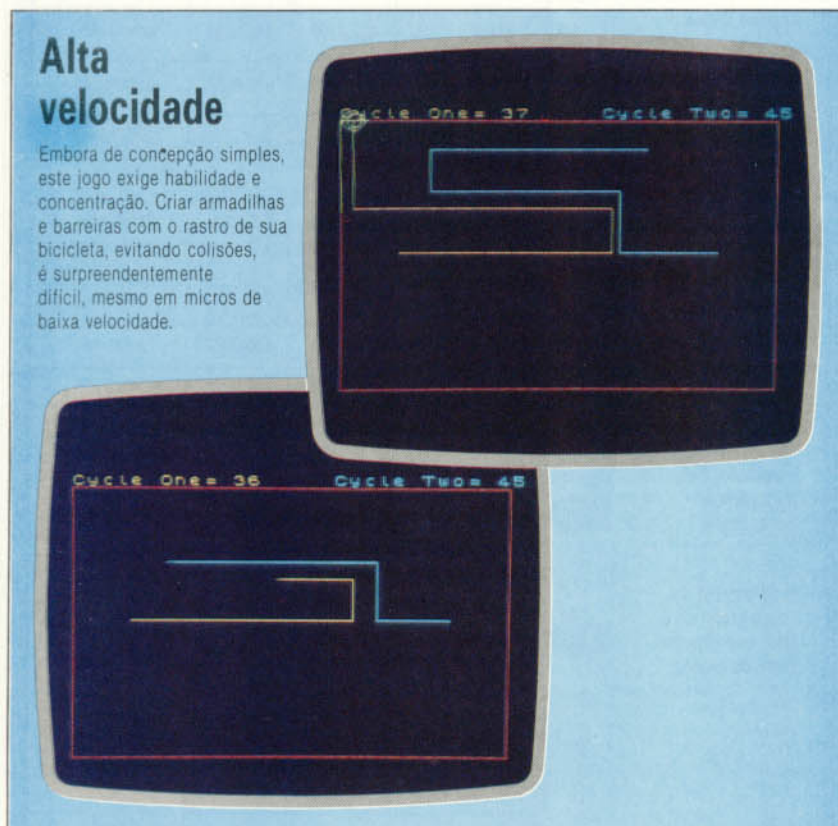
PEDALANDO

Na maioria, os jogos famosos são longos e escritos em código de máquina, devido à velocidade. Mas é possível criar um jogo em BASIC que diverte e desafia a rapidez de seus reflexos.

Baseado numa cena do filme *Tron,* produzido pelos estúdios de Walt Disney, este jogo — Pedalando — requer de você e de seu oponente muita habilidade e reações rápidas. Desenvolve-se numa arena e o único controle existente permite girar sua bicicleta num ângulo de 90° à velocidade máxima.

Os dois veículos deixam um rastro de luz e o objetivo do jogo é forçar o adversário a bater no labirinto cada vez mais estreito que você cria à medida que gira em torno da arena.

Alta velocidade

Embora de concepção simples, este jogo exige habilidade e concentração. Criar armadilhas e barreiras com o rastro de sua bicicleta, evitando colisões, é surpreendentemente difícil, mesmo em micros de baixa velocidade.

O jogo foi feito para o CP 400, cujo BASIC é lento, mas o programa está escrito em termos de velocidade e não de elegância, evitando chamadas de sub-rotinas, pois elas prejudicariam a rapidez da execução.

A primeira etapa — bastante simples — consiste no desenho da arena e do placar. O único ponto a observar é que a borda da arena fica um caractere para dentro da área útil da tela. Isso serve para assegurar que os gráficos resultantes de uma colisão com a parede da arena não saiam da tela:

```
10 P=0:Q=0:V1=0:V2=0
20 PT=-1:UM=0
100 PMODE3,1:PCLS:CLS:SCREEN0,0
110 PRINT@170,"BICICLETA 1 =";Q
120 PRINT@202,"BICICLETA 2 =";P
130 COLOR 4,1
140 LINE(1,1)-(254,190),PSET,B:
    LINE(0,0)-(255,191),PSET,B
150 REM
```

A arena foi desenhada em vermelho; o amarelo representa a bicicleta 1; e o azul, a bicicleta 2. As variáveis p e q registram a contagem do jogo para os dois competidores.

O passo seguinte consiste em inicializar todas as variáveis, e aqui é preciso pensar de que maneira vamos implementar a ação principal do jogo. A ação para uma bicicleta sozinha é simples e aparece claramente no fluxograma. Usando PPOINT, verificamos se a atual posição da bicicleta está ocupada; se estiver, passamos para a rotina de colisão. Se não estiver, passamos para aquela posição usando LINE e lemos o teclado para verificar se houve alguma mudança de direção.

A posição é então aumentada de uma unidade em nossa direção atual, e o ciclo recomeça. Portanto, precisamos de quatro variáveis: duas para as atuais coordenadas x e y e duas para nossa direção atual ao longo dos eixos x e y.

Já que se trata de duas bicicletas andando ao mesmo tempo, uma solução elegante seria usar quatro arrays de dois elementos cada, x(2) e y(2), para as posições do exemplo. Mas isso tornaria o jogo lento e, portanto, temos de usar oito variáveis separadas:

```
200 X=40: Y=88
210 M=215: N=88
220 A=1: B=0
230 I=-1: J=0
```

Isso determina as posições iniciais das bicicletas e coloca-as em movimento, uma indo em direção à outra, 1 pixel por vez. A ação básica é então simples de implementar:

```
400 IF PPOINT(X+A,Y+B)=3 OR
    PPOINT(X+A,Y+B)=4  THEN
    CO=2:GOTO700
410 IF PPOINT(M+I,N+J)=2 OR
    PPOINT(M+I,N+J)=4  THEN
    CO=3: X=M: Y=N: GOTO700
420 PSET(X,Y,2): PSET(M,N,3)
```

As linhas de 500 a 570 constituem a rotina de teclado que estabelece novos valores para A, B, I e J.

```
600 X=X+A: Y=Y+B
610 M=M+I: N=N+J
620 GOTO400
```

O único ponto que pode ser confuso é a linha 410, em que as variáveis da bicicleta 1 estão determinadas para a bicicleta 2, e uma variável nova, co, é introduzida. Isso se faz para que uma única rotina possa ser usada para a ação de colisão, onde x e y são usados simplesmente para indicar o ponto em que a colisão ocorre, e co determina a cor.

Na rotina para verificar o teclado, em vez das afirmações IF-THEN, que são lentas, podemos usar o comando rápido INKEY$ para ler as teclas escolhidas, Q e A, que controlam o movimento para cima e para baixo da bicicleta 1, e P e ENTER para a bicicleta 2. À direita e à esquerda são X e C para a bicicleta 1 e N e M para a bicicleta 2.

```
500 T$=INKEY$:IF T$="A" THEN A=0:
    B=1
510 IF T$="Q" THEN A=0: B=-1
520 IF T$="X" THEN A=-1: B=0
530 IF T$="C" THEN A=1: B=0
540 IF T$=CHR$(13) THEN I=0: J=1:
    UM=1
550 IF T$="P" THEN I=0: J=-1: UM=1
560 IF T$="M" THEN I=1: J=0: UM=1
570 IF T$="N" THEN I=-1: J=0: UM=1
```

Resta agora a rotina de colisão e atualização do placar. Escolhe-se uma série expansiva de círculos concêntricos, centralizados no ponto de impacto, com raios de 4, 6 e 8 pixels:

```
700 FOR D=1 TO 3
710 CIRCLE(X,Y),2+D*2,CO
720 NEXT D
730 IF CO=2 THEN P=P+1: GOTO750
740 Q=Q+1
750 FOR Z=1 TO 200: NEXTZ: GOTO100
```

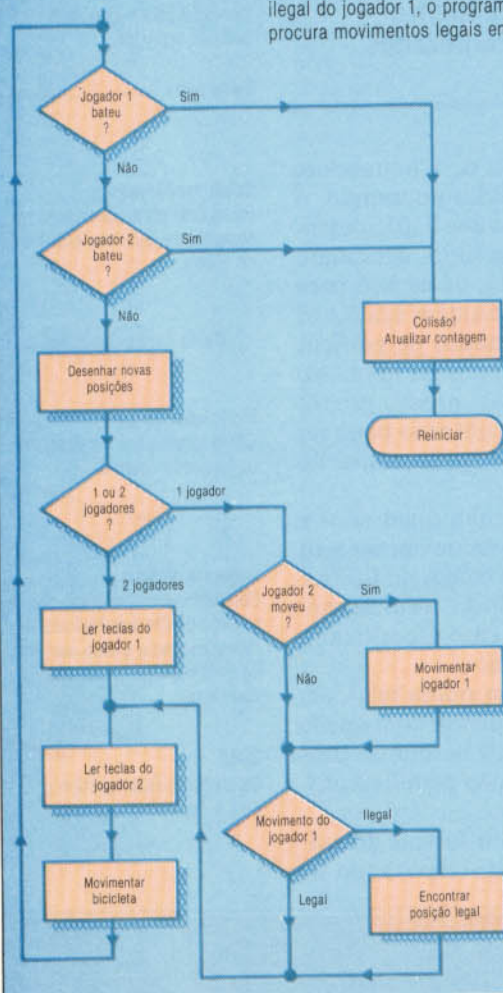
Isso encerra o jogo, com a última afirmação voltando em loop para os procedimentos iniciais. No entanto, o jogo poderia começar com um procedimento inicial que o tornaria mais interessante:

```
300 T$="":PRINT@448,"USE QUALQUER
    TECLA PARA  INICIAR";
310 T$=INKEY$: IF T$="" THEN 310
320 T$="":SCREEN 1,0
```

Isso lhe dá um intervalo entre rodadas consecutivas. Agora, só falta gravar o programa usando SAVE (para disco) ou CSAVE (para cassete). Introduzindo-se a linha 5 RUN, o jogo será executado automaticamente assim que for carregado.

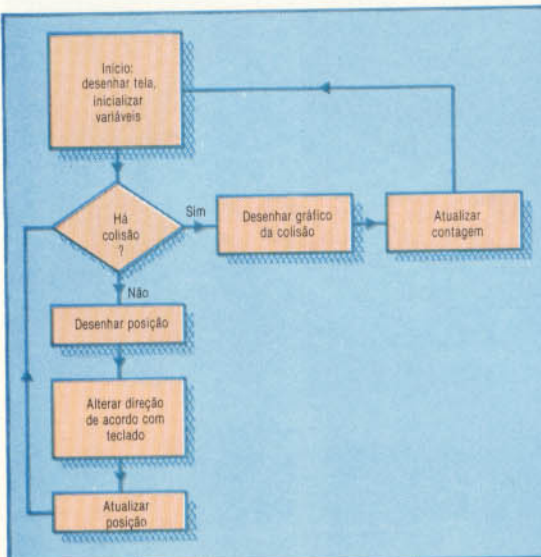
O jogo se tornaria mais excitante com telas de instruções, telas de carga, opção para um só jogador com uma rotina estratégica para controlar a outra bicicleta, som e gráficos mais complexos. Certamente, porém, essas alternativas viriam em prejuízo da rapidez da ação.

Para um só jogador

Encaixando
Na versão individual, o computador é o jogador 1. Ignora-se a parte do programa que verifica as teclas de comando do jogador 1, e o algoritmo no código dado permite que o jogador 1 continue a andar em linha reta até que isso gere um movimento ilegal, ou que o jogador 2 ande. Neste último caso, o movimento é duplicado e sua legalidade é verificada. Quando se gera um movimento ilegal do jogador 1, o programa procura movimentos legais em ângulos retos com a direção do movimento ilegal; se nenhum for encontrado, então a posição é crítica. Há muitas maneiras de implementar uma versão desse jogo para um só jogador. As modificações que sugerimos possibilitam escolher entre as versões para um e dois jogadores no começo de cada partida. Não é difícil obter algoritmos satisfatórios para este jogo, mas há dificuldades para implementá-los em BASIC sem diminuir consideravelmente a velocidade. No programa, eis as modificações:

```
260 PRINT@0,"NO. DE
    JOGADORES (1/2)"I
270 INPUT TE$
280 IF TE$="2" THEN K=2
    ELSE K=1:X=70
```

Com estas linhas, o usuário pode escolher os tipos de jogo e implementar essa escolha tendo acesso ao estratagema de um jogador, entre as linhas 440 e 460, ou à versão padrão para dois jogadores, entre as linhas 500 e 570. Nossa estratégia está contida nestas linhas:

```
430 ON K GOTO 435,500
435 T$=INKEY$:T1=RND(2):
    IF T1=1 THENPT=1
    ELSE PT=-1
440 IF UM=PT THEN A=PT*J:
    B=PT*I: UM=0
450 IF PPOINT(X+A,Y+B)=3
    OR PPOINT(X+A,Y+B)=4
    THEN GOTO 540
455 T1=RND(2): IFT1=1
    THEN PT=1 ELSE PT=-1
460 A=A*PT: B=B*PT: D=A:
    A=B: B=D: IF PPOINT
    (X+A,Y+B)=3 OR PPOINT
    (X+A,Y+B)=4 THEN
    GOTO540
490 A=-A: B=-B: GOTO 540
```

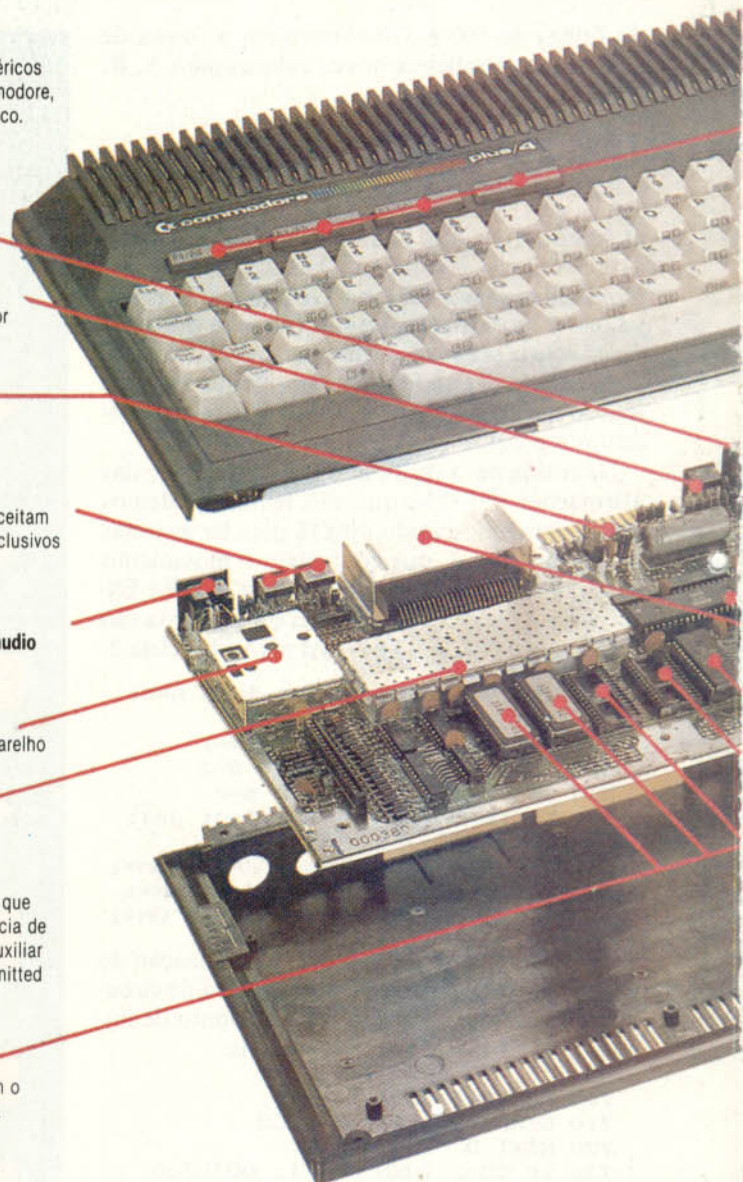
Opção para um só jogador
Este fluxograma simplificado mostra a estrutura do programa com apenas um jogador. Cada processo é repetido no jogo completo para dois jogadores.

PLUS/4

A Commodore submeteu o modelo 64 a aperfeiçoamentos e lançou o Plus/4, que dispõe de uma versão melhorada de BASIC, quatro programas aplicativos incorporados e 64 Kbytes de memória.

Modelo comercializado depois do Commodore 64, um dos micros mais vendidos no mundo, o Plus/4 utiliza o microprocessador 7501, desenvolvido a partir do 6502. Esse chip, acessando mais de 64 Kbytes de memória, dá espaço para um BASIC potente, enquanto mantém sua RAM livre para o usuário. Os comandos de gráficos e som são particularmente dignos de nota. No modo gráfico, o comando DRAW produz pontos ou linhas, e qualquer tipo de contorno pode ser preenchido com cores, por meio do comando PAINT.

Outro comando, BOX, desenha quadrados e retângulos preenchidos com cores ou apenas seus contornos. Por fim, o versátil comando CIRCLE possibilita traçar círculos ou elipses, além de simples arcos, bastando em cada caso especificar a altura e o comprimento da curva.

Todos os comandos operam numa tela com resolução de 320 x 200 pixels, porém com opção para 121 cores, criadas a partir de quinze tons básicos. No entanto, o Plus/4 não permite a produção de sprites.

Os sons são emitidos pelo alto-falante do monitor de vídeo, em dois canais, respeitando os

O esperado sucessor do Commodore 64 possui todas as características que estão se tornando padrão na última geração de microcomputadores: o agrupamento das teclas do cursor, a memória de grande capacidade e os aplicativos incorporados. Claro sinal de que a Commodore não pretende perder sua posição de liderança neste competitivo mercado.

Teclas de função
Cada uma das quatro teclas pode ser programada com dois comandos de até 128 caracteres.

Teclas do cursor

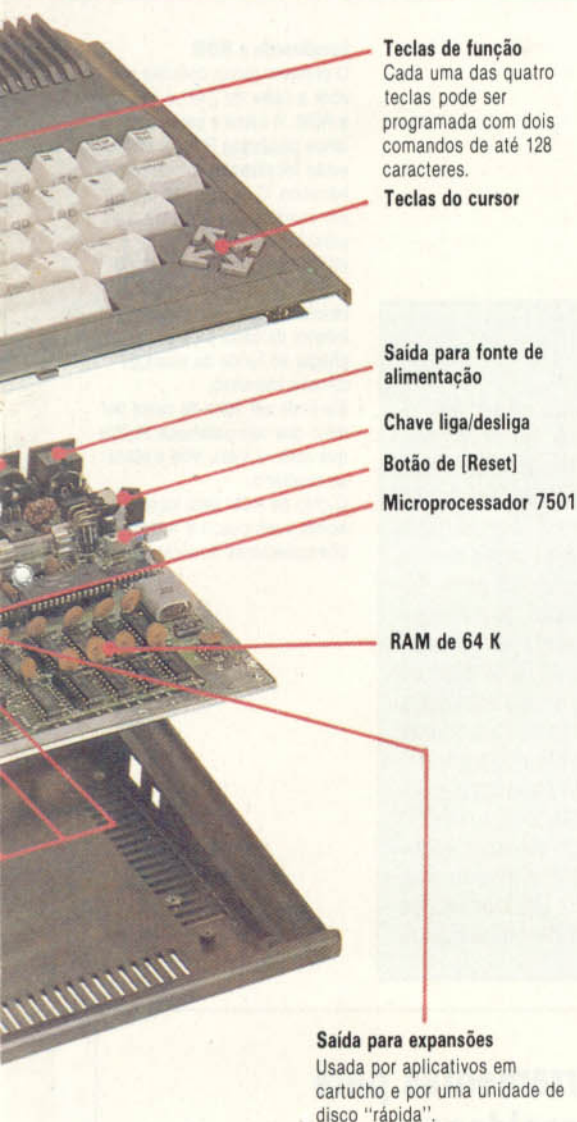
Saída para fonte de alimentação

Chave liga/desliga

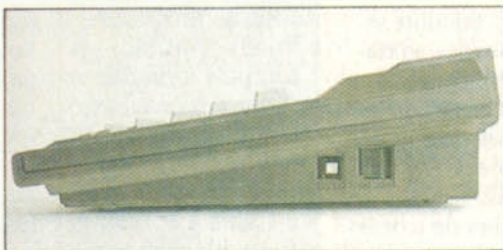
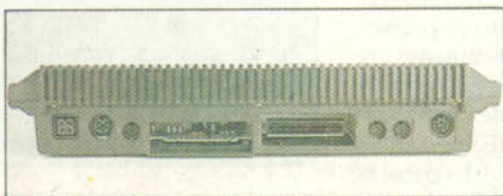
Botão de [Reset]

Microprocessador 7501

RAM de 64 K

Saída para expansões
Usada por aplicativos em cartucho e por uma unidade de disco "rápida".

Teclas de função

Teclas do cursor

Botão de [Reset]

Saída para expansões

Inovações
Ao contrário dos modelos antigos da Commodore, o Plus/4 possui as teclas [Escape] e [Reset]. Outras inovações são as teclas para movimentação do cursor agrupadas, as teclas de função (programáveis em BASIC) e a tecla [Help]. As saídas para fonte de alimentação, cassete, joysticks e expansões são todas diferentes das encontradas no Commodore 64 e Vic-20.

COMMODORE PLUS/4

MICROPROCESSADOR
7501.

CLOCK
0,9 ou 1,8 MHz.

MEMÓRIA
64 K de ROM, 64 K de RAM.

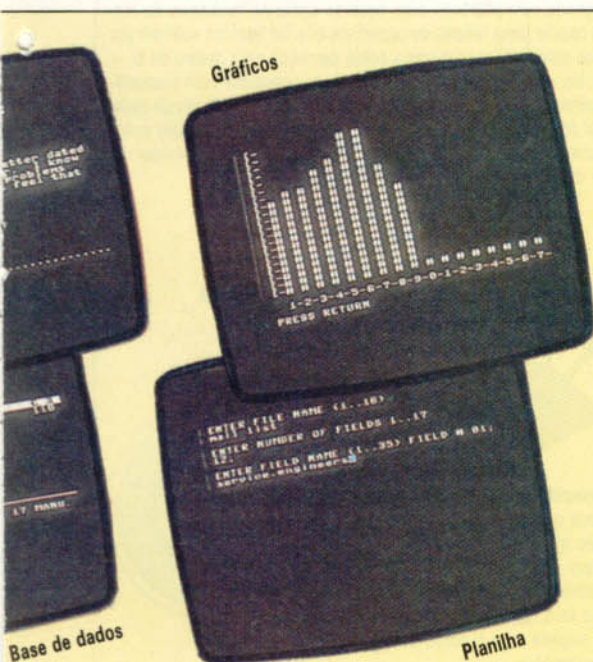
SISTEMA OPERACIONAL
DOS.

VÍDEO
Saída para monitor e TV comum.
Modo texto: 25 linhas de quarenta caracteres.
Modo gráfico: 320 x 200 pixels, em 121 cores.

TECLADO
Tipo QWERTY, com 67 teclas de função e quatro teclas de cursor, agrupadas em forma de diamante.

LINGUAGENS
BASIC.

PERIFÉRICOS
Interface serial e paralela. Saídas para joysticks, impressora matricial — em preto e branco ou em cores —, impressora margarida e plotter em quatro cores.

DOCUMENTAÇÃO
Manuais razoáveis que ensinam a programar e a usar os aplicativos.

Custo oculto
O software incorporado à ROM do Plus/4 inclui um processador de texto, planilha eletrônica, banco de dados e gerador de gráficos. Apesar da grande vantagem de estarem integrados, esses aplicativos só podem ser implementados com uma unidade de disco, o que aumenta bastante o custo da máquina.

controles que não fogem ao padrão normal. O maior aperfeiçoamento reside no corpo central da linguagem BASIC: comando AUTO, para produzir linhas numeradas automaticamente; RENUMBER, para renumerar as linhas do programa; e VERIFY, para conferir se o programa foi gravado no cassete ou no disco.

No modo texto, o Plus/4 mostra no vídeo 25 linhas de quarenta caracteres. O usuário pode especificar dois pontos na tela a fim de marcar as margens de uma "janela", dentro da qual vão aparecer todos os textos.

A quantidade de memória permite que o Plus/4 tenha quatro programas aplicativos incorporados no chip da ROM, mas que exigem a utilização de disquetes: processador de texto, planilha eletrônica, banco de dados e gerador de gráfico — planejados para execução integrada.

Muitos usuários ficarão mais entusiasmados com o Tedmon — o programa monitor em código de máquina — do que com os aplicativos, pois ele é de grande utilidade para se programar diretamente em linguagem de máquina.

A TROCA DO CHIP

Saber montar cabos e conectores é fundamental em eletrônica. Convém, portanto, aprender a dessoldar, mudar componentes e remover um chip de ROM, substituindo-o por um soquete para facilitar sua troca.

Depois de aprender a soldar, o passo seguinte será descobrir como desfazer as conexões soldadas, de modo bem-acabado e sem confusão. Quando estamos lidando só com cabos e conectores, não precisamos preocupar-nos demais — na certa será mais simples cortar o cabo, jogar fora o conector e substituí-lo.

Mas o que acontece quando se trata de trocar ou mudar de lugar um chip? Mesmo os transistores simples têm três terminais. Para remover um transistor de sua placa de circuito impresso, precisamos aquecer os três simultaneamente até o ponto de fusão da solda, de modo que o transistor possa ser liberado; ou temos de tirar a solda de cada um dos pinos por vez. E, se a idéia de desligar três pinos de uma vez já é desencorajadora, o que dizer dos quarenta pinos do processador típico de 8 bits?

Extração da ROM

Em vez de escolher um chip ao acaso para ser substituído (ou acrescentar chips de RAM para expandir a capacidade de uma máquina), vamos executar um projeto um pouco mais ambicioso: substituir a ROM do BASIC do ZX81 por um soquete — ligando um cabo multivias entre as conexões originais do chip e um pedaço de Protoboard, onde instalamos o soquete para receber a ROM que extraímos. A razão de escolhermos o ZX81 como objeto deste exercício está no fato de que o BASIC pode ser substituído pelo FORTH como linguagem residente (em ROM). Além da nova linguagem, a ROM do FORTH também incorpora um sistema operacional multitarefa, permitindo que mais de um programa rode ao mesmo tempo, cada qual inteiramente independente dos demais. Essa é uma conquista notável para uma máquina tão pequena, embora ela necessite de um mínimo de 2 Kbytes de RAM, o que pode exigir o acréscimo de um módulo de memória.

Além de ser um exercício das técnicas que até agora examinamos, um pequeno projeto como este possibilita o treino na manipulação de componentes e mostra a você como é necessário ser cuidadoso e preciso. Para testar o serviço pronto, recomenda-se o uso de um multímetro.

Receita

Se um chip tiver de ser substituído ou reposicionado, ou se for necessário colocar um novo chip, será indicada a instalação de um soquete. Os chips com soquete podem ser substituídos em pouco tempo. Há à venda sofisticados suportes para chips, chamados ZIF (Zero Insertion Force), que reduzem ao mínimo o risco de amassar um pino. Você precisará também de um carretel de mecha e de outras ferramentas, para a dessoldagem. O exercício que escolhemos é a retirada da ROM BASIC do ZX81 e sua substituição por um soquete padrão, do qual será puxado um cabo plano de fora da máquina para o soquete. O objetivo será substituir o BASIC do ZX81 pelo FORTH-in-ROM multitarefa, da David Husband — dando ao usuário, ao mesmo tempo, a opção de posteriormente retornar ao BASIC. Se você quiser trabalhar acompanhando este exemplo, precisará de pedaços de Protoboard, de cabo de 28 fios e de um pedaço de poliestireno.

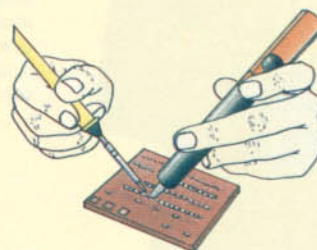
Localizando a ROM

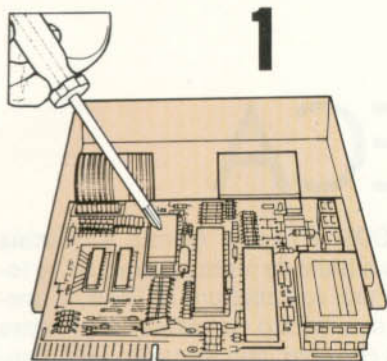
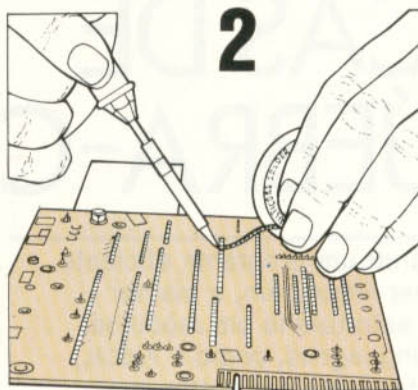
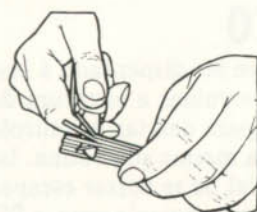
O primeiro passo consiste em abrir a caixa do ZX81 e localizar a ROM. A caixa é presa por cinco parafusos Philips; três estão localizados sob os pés de borracha. O pé que não contém um parafuso é o que está mais próximo aos soquetes EAR e MIC. Levante com cuidado os outros três e retire os parafusos. Levante a face inferior da caixa para chegar ao fundo da placa de circuito impresso.

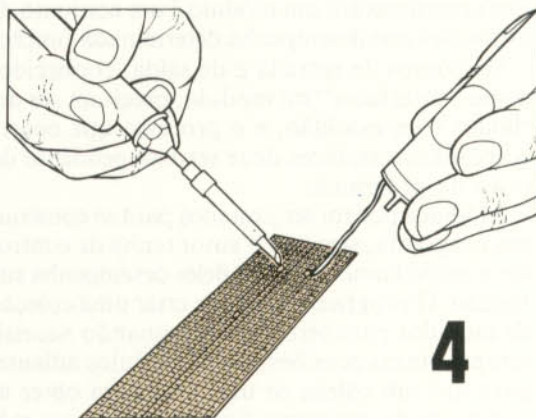
Ela pode ser solta da caixa por meio dos três parafusos Philips que estão à vista. Vire a placa ao contrário.

O chip da ROM está localizado acima e um pouco à esquerda dos conectores do teclado.

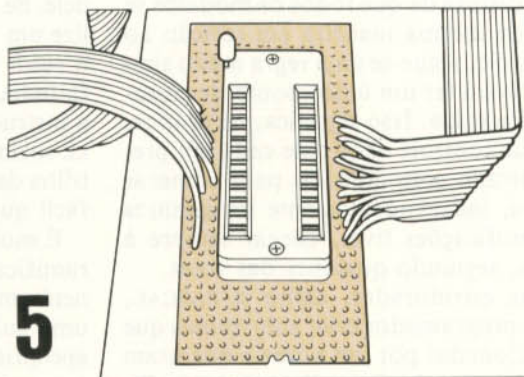
Ferramentas para dessoldagem

Há dois instrumentos patenteados para a dessoldagem: a "mecha para solda" — um fio de cobre muito fino, trançado numa fita e impregnado com solvente, cuja ação se apóia no fenômeno da função capilar (uma tensão de superfície que faz líquidos subirem por tubos estreitos) para puxar a solda derretida para dentro de si — e outro, considerado mais adequado, que consiste num dispositivo parecido com uma bomba de bicicleta, sugando em vez de expelir ar pela ponta. Esse instrumento opera com mais rapidez que a mecha, além de ter durabilidade bem maior. É essencial usar um dos dois para remover componentes.

A dessoldagem

Aqueça com o soldador a junção até a solda derreter. Aplique a ferramenta de dessoldagem, pressione o botão acionador, e a solda será sugada para o corpo da ferramenta.

Retirando a ROM
Na dessoldagem, como na execução de junções, o segredo está na aplicação de calor suficiente. Com a ponta do soldador, coloque a mecha sobre a junção até que o líquido escorra. Você verá a mecha sugar a solda. Corte a extremidade da mecha de vez em quando, pois cada pequena parte fica logo saturada.

3

Cabo plano
Você vai usar o cabo multivias ou paralelo de 28 fios (ou dois com catorze fios). Descubra 1 cm da ponta de cada um dos 28 fios e estanhe cada um deles. Trabalhando a partir da ponta marcada com um pequeno recorte semicircular, solde o cabo plano no lugar, ao longo de ambos os lados. Todos os chips têm esses recortes, ou às vezes pontos para indicar o modo como devem ser alinhados.

4

Fazendo conexão
Corte os dois pedaços de cabo plano no tamanho correto, descasque e estanhe os 28 fios. Solde-os no Protoboard, assegurando-se de que os 28 fios da ponta marcada na área que mantinha o chip serão ligados na mesma ponta do soquete. Nenhum dos fios pode ficar cruzado.

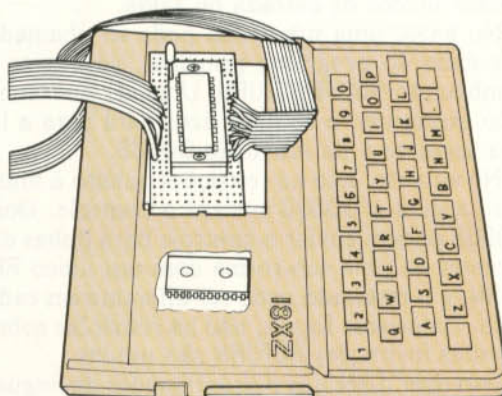
Protoboard
Este produto é projetado para a montagem de protótipos: é como uma placa de circuito impresso, com muitas carreiras de furos, ligados por trilhas de cobre. Cortando essas trilhas, criam-se conexões entre os componentes colocados na placa. É ideal para experiências e para montagens que normalmente exigem muitas correções.

5

Montando o soquete
Para esta finalidade, o Protoboard é o meio mais apropriado. Corte um pedaço de quinze pontos de contato de largura por 25 cm de comprimento. Coloque o soquete do chip de modo que ele enquadre o isolamento central; solde um pino de canto para prendê-lo firmemente. Então trabalhe um lado de cada vez.

6

Acabamento
Quando todas as conexões estiverem feitas, verifique cada uma delas, tanto na placa do computador como na nova, para assegurar-se de que estão separadas. Ao trabalhar um componente como este, com muitas conexões próximas, a possibilidade de curto-circuito é grande. Verifique as conexões com uma lente de aumento e, se estiver em dúvida, raspe entre os vãos com um estilete ou outro instrumento de metal.

AVISO

A garantia de seu computador — se ainda estiver em vigor — poderá ser cancelada se pessoa não autorizada abrir a máquina.

PEÇAS DE QUEBRA-CABEÇA

Para construir um programa eficiente em qualquer linguagem, a melhor maneira é estruturá-lo em módulos. Algumas linguagens, como a PASCAL, facilitam isso, mas em outras é preciso usar artifícios.

Em programação, um módulo é um conjunto de instruções que desempenha determinada função.

Os pontos de entrada e de saída, conhecidos como "interfaces" do módulo, precisam ser definidos com exatidão, e o processo que ocorre entre essas interfaces deve ser independente do resto do programa.

Módulos podem ser juntados para se construir um programa, sem que o autor tenha de controlar o modo como cada um deles desempenha sua função. O programador pode criar uma coleção de módulos para serem usados quando necessitar; em outras ocasiões, passa módulos adiante, para que um colega os use. Mas para obter as vantagens da programação estruturada em módulos precisamos tomar nota do fluxo de controle e do fluxo de dados ao escrever os módulos.

Para ter a certeza de que todos os módulos se comportam da mesma maneira em relação ao fluxo de controle, segue-se uma regra muito simples: todos devem ter um único ponto de entrada e uma única saída. Isso significa, na prática, que o fluxo de controle dentro de cada um precisa ser projetado com cuidado para começar num ponto e, independentemente de quantos loops ou ramificações tiver, chegar sempre à mesma saída, seguindo qualquer das rotas.

Linguagens estruturadas, como a PASCAL, permitem ao programador criar sub-rotinas que podem ser acionadas por um nome e que usam suas próprias variáveis. Essas linguagens facilitam ao programador a entrada ou saída de uma rotina (chamada "procedimento") por meio dos pontos únicos de entrada ou saída.

No BASIC uma sub-rotina pode ser chamada por meio do programa principal, usando-se a combinação GOSUB-RETURN. Uma vez que tenha sido executada, o controle retornará para a linha seguinte à da instrução GOSUB.

No entanto, não há restrições quanto à linha para a qual o GOSUB enviará o controle. Dois GOSUB podem enviar o controle para linhas diferentes de uma sub-rotina com um único RETURN, e o resultado pode ser diferente em cada caso. Da mesma forma, não há restrições sobre quantas instruções RETURN são usadas.

Isso quer dizer que o programador de linguagem BASIC deve autodisciplinar-se, verificando se todos os GOSUB para a mesma sub-rotina apontam para a linha de mesmo número e se toda sub-rotina tem somente um RETURN. O melhor é adquirir o hábito de marcar a primeira linha de todas elas com uma instrução REM, dando a cada qual um título, e usar aquela linha como ponto de entrada. Faça do RETURN a última linha de sub-rotina.

A regra do GOTO

Um cuidado especial deve ser dispensado à instrução GOTO, que pode arruinar a estrutura de um programa. Só a use para desviar o controle para uma linha dentro da mesma sub-rotina. Isso evita o perigo potencial de se deixar escapar um RETURN ou de passar o controle para o RETURN errado. Há ocasiões em que é necessário não executar algumas linhas de uma rotina. Nesse caso, dê um GOTO para a linha onde está o RETURN e não haverá problemas.

Usar o GOTO dentro de um loop é ainda mais perigoso. Se o controle sai do loop, o BASIC não sabe disso e presume que o resto do programa é o corpo daquele loop. Quando estiver num loop, nunca dê um GOTO para uma linha fora dele. Se um loop precisa terminar mais cedo, utilize um contador ou uma variável de teste para o valor final e dê um GOTO para linha de teste (a linha com o NEXT ou o WHILE). Como com a instrução RETURN, coloque o NEXT ou o WHILE sozinhos numa linha, para facilitar. Seguir a trilha da estrutura de um programa é muito mais fácil quando se evitam os GOTO.

É muito comum o controle se extraviar pelas ramificações. Assim, procure não permitir que nenhuma decisão envie o controle para fora de uma sub-rotina, a menos que seja chamando apropriadamente outra sub-rotina. Lembre-se de que cada uma delas tem um único ponto de saída e, então, certifique-se de que é possível seguir o fluxo de controle através de cada ramificação até aquele ponto. Desenhar um fluxograma para a rotina torna mais fácil a checagem desse aspecto. Colocar uma sinalização (flag) pode reduzir a necessidade de GOTO em rotinas que envolvem loops e ramificações.

Para que os módulos possam ser utilizados como independentes uns dos outros, você deve projetá-los de maneira que a única influência de um sobre o outro seja por intermédio dos dados que trafegam entre eles. O programa principal passa os dados a um módulo e, quando este tiver sido executado, qualquer resultado pode ser passado de volta.

Os dados circulam pelos programas dentro de variáveis e a liberdade de movimento de uma va-

riável é chamada "raio de ação". Muitas linguagens de programação restrigem o raio de ação de uma variável a determinada sub-rotina. Em PASCAL as variáveis usadas em determinada sub-rotina (procedimento) devem ser declaradas para aquele procedimento. Variáveis declaradas para o programa principal são globais e são usadas em qualquer lugar do programa (inclusive dentro de qualquer de seus módulos). Variáveis declaradas num procedimento específico, contudo, são locais e só podem ser usadas dentro dele.

As variáveis locais, às vezes, têm o mesmo nome das globais, e usar uma delas não afeta o valor de outra. Uma linguagem que emprega variáveis locais permite escrever sub-rotinas sem preocupação quanto à influência que as variáveis de uma terão sobre as de outras.

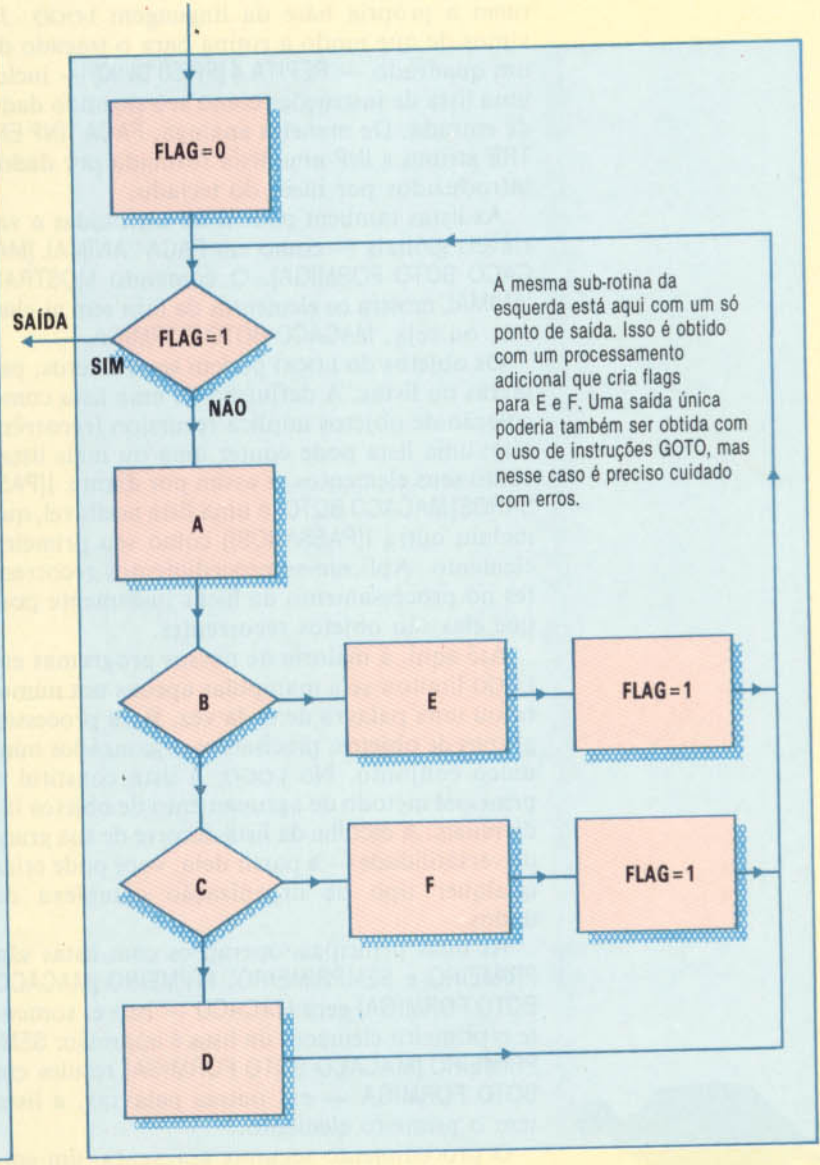
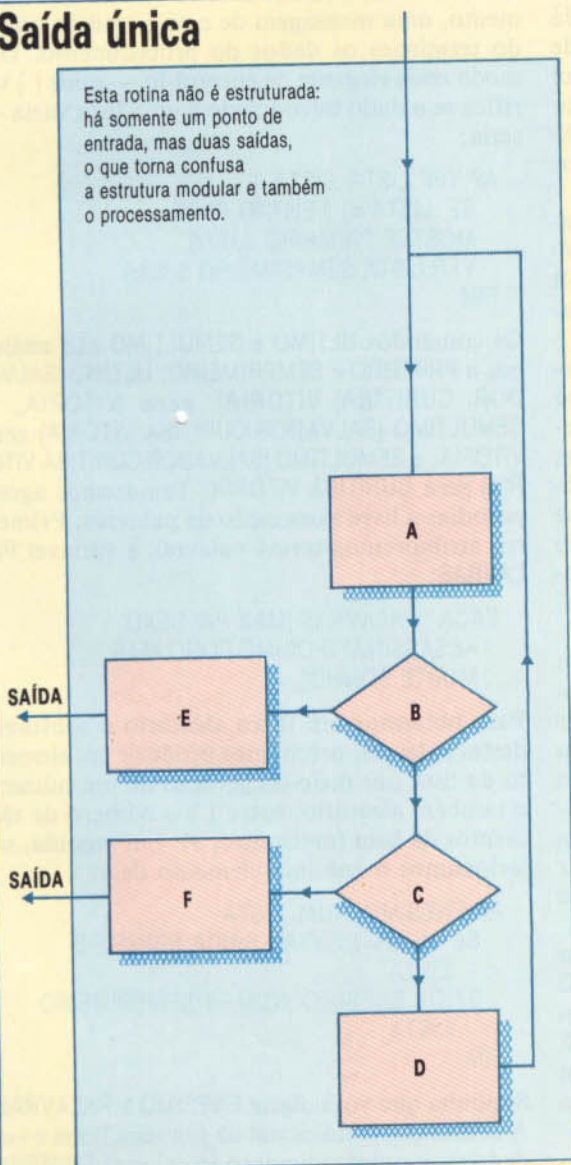
Poucas versões da linguagem BASIC adotam variáveis locais, o que significa que, se quisermos escrever sub-rotinas independentes, devemos de alguma forma simular variáveis locais.

A maneira mais simples de conseguir isso é adotar nomes identificadores de variáveis que desempenham tarefas diferentes. Algumas convenções já existem e são usadas por programadores. Usar I, J e K como contadores de loops e valores de índices é muito comum.

Descrito o programa por um fluxograma, fica fácil numerar as sub-rotinas ou dar a elas outro tipo de identificação. Qualquer variável global que precise ser transformada em local em determinada sub-rotina ganha, então, como sufixo, o código correspondente à sub-rotina, para torná-la única. Assim, a rotina número 5 pode utilizar a variável local SOMA5 e TOTAL5, para distingui-las de SOMA12 e TOTAL12 na rotina número 12. Cuidado com a possibilidade de o BASIC que você está usando considerar somente os dois primeiros caracteres. As variáveis usadas para passar valores entre sub-rotinas e as usadas apenas no programa principal não precisam ser codificadas assim.

Saída única

Esta rotina não é estruturada: há somente um ponto de entrada, mas duas saídas, o que torna confusa a estrutura modular e também o processamento.

A mesma sub-rotina da esquerda está aqui com um só ponto de saída. Isso é obtido com um processamento adicional que cria flags para E e F. Uma saída única poderia também ser obtida com o uso de instruções GOTO, mas nesse caso é preciso cuidado com erros.

POESIA ALEATÓRIA

Você imagina a tartaruga escrevendo poemas? Isso é o que conseguiremos que ela faça ao estudar as listas, um recurso essencial do LOGO. Antes, porém, vamos rever a recorrência e a livre associação de palavras.

Uma lista não passa de uma coleção ordenada de objetos. No LOGO, as listas sempre vêm escritas entre colchetes — por exemplo, [CURITIBA SALVADOR VITORIA]. Várias vezes nós as manipulamos nesta série de artigos, pois elas constituem a própria base da linguagem LOGO. Já vimos de que modo a rotina para o traçado de um quadrado — REPITA 4 [FR 50 DI 90] — inclui uma lista de instruções como seu segundo dado de entrada. De maneira análoga, FACA "INP ENTRE atribui a INP uma lista formada por dados introduzidos por meio do teclado.

As listas também podem ser atribuídas a variáveis globais — como em FACA "ANIMAL [MACACO BOTO FORMIGA]. O comando MOSTRAR :ANIMAL mostra os elementos da lista sem as chaves, ou seja, MACACO BOTO FORMIGA.

Os objetos do LOGO podem ser números, palavras ou listas. A definição de uma lista como coleção de objetos implica recursion (recorrência): uma lista pode conter uma ou mais listas como seus elementos, e assim por diante. [[PASSAROS] MACACO BOTO] é uma lista aceitável, que incluiu outra ([PASSAROS]) como seu primeiro elemento. Aplicam-se procedimentos recorrentes no processamento de listas justamente porque elas são objetos recorrentes.

Até aqui, a maioria de nossos programas em LOGO limitou-se a manipular apenas um número ou uma palavra de cada vez. Para processar grupos de objetos, precisamos organizá-los num único conjunto. No LOGO, a lista constitui o principal método de agrupamento de objetos individuais. A escolha da lista decorre de sua grande versatilidade — a partir dela, você pode criar qualquer tipo de organização complexa de dados.

As duas principais operações com listas são PRIMEIRO e SEMPRIMEIRO. PRIMEIRO [MACACO BOTO FORMIGA] gera MACACO — isto é, somente o primeiro elemento da lista é impresso. SEMPRIMEIRO [MACACO BOTO FORMIGA] resulta em BOTO FORMIGA — em outras palavras, a lista sem o primeiro elemento.

O procedimento seguinte apresenta, um embaixo do outro, todos os elementos da lista:

```
AP VER.LISTA :LISTA
  MOSTRE PRIMEIRO :LISTA
  VER.LISTA SEMPRIMEIRO :LISTA
FIM
```

Assim, VER.LISTA [SALVADOR CURITIBA VITORIA] imprime:

```
SALVADOR
CURITIBA
VITORIA
```

O comando inicial imprime o primeiro elemento da lista e depois transfere, para outra cópia de VER.LISTA, a tarefa de apresentação do resto da lista de entrada. Ao se executar esse procedimento, uma mensagem de erro é emitida quando terminam os dados do procedimento. Um modo mais elegante de encerrá-lo — onde [ ] verifica se o dado introduzido é uma lista vazia — seria:

```
AP VER.LISTA :LISTA
  SE :LISTA = [ ] ENTAO PARE
  MOSTRE PRIMEIRO :LISTA
  VER.LISTA SEMPRIMEIRO :LISTA
FIM
```

Os comandos ULTIMO e SEMULTIMO são análogos a PRIMEIRO e SEMPRIMEIRO. ULTIMO [SALVADOR CURITIBA VITORIA] gera VITORIA, e SEMULTIMO [SALVADOR CURITIBA VITORIA] gera VITORIA, e SEMULTIMO [SALVADOR CURITIBA VITORIA] gera CURITIBA VITORIA. Tentaremos agora parodiar a livre associação de palavras. Primeiro, atribuiremos certas palavras à variável PALAVRAS:

```
FACA "PALAVRAS [MAE PAI SEXO
  ASSASSINATO CIUME FOGO MAR
  MORTE SONHO]
```

Para obtermos um fluxo aleatório e constante destas palavras, precisamos produzir um elemento da lista por meio da geração de um número n também aleatório, entre 1 e o número de elementos da lista (neste caso, 9). Em seguida, selecionamos o enésimo elemento dela.

```
AP ENESIMO :NUM :LISTA
  SE :NUM = 1 ENTAO SAIDA PRIMEIRO
    :LISTA
  SAIDA ENESIMO :NUM - 1 SEMPRIMEIRO
    :LISTA
FIM
```

Suponha que você digite ENESIMO 1 :PALAVRAS. A declaração condicional na primeira linha é verdadeira, e o procedimento imprimirá PRIMEIRO :PALAVRAS, ou seja, MAE.

Experimente em seguida ENESIMO 2 :PALAVRAS — agora a declaração condicional é falsa, e o procedimento passa a ENESIMO 1 SEMPRIMEIRO :PALAVRAS. O primeiro elemento da lista é ignorado, e o procedimento escreve a primeira palavra dentre os elementos restantes — PAI. Assim, o procedimento para a geração de uma palavra aleatória a partir daquele vocabulário seria:

```
AP GERAR :LISTA
  SAIDA ENESIMO ((SORTEIE 9) + 1) :LISTA
FIM
```

Para rodá-lo, digite GERAR :PALAVRAS.

Esse procedimento restringe-se a uma lista de nove elementos. Se quisermos definir o número de itens que compõem determinada lista, usamos a seguinte rotina:

```
AP TAMANHO :LISTA
  SE :LISTA = [ ] ENTAO SAIDA 0
  SAIDA 1 + TAMANHO SEMPRIMEIRO :LISTA
FIM
```

Digite TAMANHO [FICCAO CIENTIFICA] para entender seu funcionamento. Como a lista contém várias palavras, a primeira declaração condicional é falsa, levando o procedimento a mostrar 1 + TAMANHO [FICCAO]. Agora o resultado de TAMANHO [FICCAO] é 1 + TAMANHO [ ]. Quando se chama TAMANHO e se introduz [ ], a declaração condicional da primeira linha torna-se verdadeira e o procedimento resulta em 0. Desta vez, TAMANHO [FICCAO] gera 0 + 1 = 1 e, por fim, TAMANHO [FICCAO CIENTIFICA] gera 1 + 1 = 2. Um procedimento mais geral para a geração de palavras aleatórias a partir de listas de qualquer tamanho pode ser:

```
AP GERAR :LISTA
  SAIDA ENESIMO ((SORTEIE TAMANHO
    :LISTA) + 1) :LISTA
FIM
```

Se você quiser uma seleção de dez comentários, basta digitar:

```
REPITA 10 [MOSTRE GERAR :PALAVRAS]
```

Esses programas para o processamento de listas possuem estrutura semelhante à de vários procedimentos para desenhos que vimos antes:

- Quando a tarefa é muito simples, realize-a e pare.
- Caso contrário, complete apenas pequena parte da tarefa.
- A seguir, transfira o restante da tarefa para outro procedimento (em geral, uma cópia do procedimento original).

Esta estratégia é de grande eficácia, sendo freqüentemente utilizada nos programas de processamento de listas. Compare um programa para o traçado de um quadrado "cheio" (totalmente preenchido com linhas), como este:

```
AP QUADRADO :N
  SE :N = 0 ENTAO PARE
  REPITA 4 [FR :N DI 90]
  QUADRADO :N - 1
FIM
```

com a versão de LISTAR já apresentada. Observe a perfeita equivalência das estruturas dos dois procedimentos.

Quando não é possível a utilização da condicional na primeira instrução, o programa será executado indefinidamente, pois não existirá o comando primitivo PARE:

```
AP POLI :N
  FR 30 DI (360/ :N)
  POLI :N
FIM
```

O programa acima desenha um polígono com um número desejado de lados. Para parar a execução, utilize as teclas [CTRL] - G no MLOGO.

Poesia aleatória

Para escrever alguns poemas, precisamos gerar frases completas e não apenas palavras isoladas.

```
AP POEMA :TAMANHO
  SE :TAMANHO = 0 ENTAO MOSTRE ". PARE
  MOSTRAR"' '
  MOSTRAR GERAR :PALAVRAS
  POEMA :TAMANHO - 1
FIM
```

MOSTRAR "' ' serve para garantir o espaço entre as palavras. Para produzir uma frase de seis palavras com este procedimento, digite POEMA 6.

Um dos recursos para se ampliar a lista inicial de palavras, sem reescrevê-la, é a operação SENTENCA. Ela aceita a introdução de dois dados e os transforma numa lista. Dessa maneira, SENTENCA "GELEIA [POTE DE MEL] gera [GELEIA POTE DE MEL].

```
AP SOMA.PALAVRAS :LISTA
  FACA "PALAVRAS SENTENCA :LISTA
:PALAVRAS
  FIM
```

Agora podemos ampliar PALAVRAS com SOMA.PALAVRAS [ANSIEDADE REPRESSAO [MEDO DE VOAR]]. Resta ainda a questão de saber se não se atribuiu, anteriormente, algum valor à variável PALAVRAS. O primitivo VALOR? verifica se isso ocorreu, respondendo VERD no caso de seu dado de entrada estar associado a algum valor:

```
AP SOMA.PALAVRAS :LISTA
  SE VALOR? :PALAVRAS ENTAO FACA
    "PALAVRAS [ ]
  FACA "PALAVRAS SENTENCA :LISTA
    :PALAVRAS
FIM
```

Empregando outra lista de palavras, criamos com esse procedimento o seguinte poema:

```
ESPECTRO RUIDOSAMENTE FALOU
ESPLENDIDO PARANOICO PLANETA
ATERRORIZADO O COM VERDE ESPECTRO
FLUTUANDO PARANOICO ROBO HOMEM VOOU
FALOU FLUTUANDO RUIDOSAMENTE
```

Uma falha óbvia deste poema é sua total desconsideração pela gramática. Talvez seja possível

Simplificação

ENTRE	
FACA	
PRIMEIRO	
SEMPRIMEIRO	SP
ULTIMO	
SEMULTIMO	SU
SENTENCA	SN
MOSTRE	MO

torná-lo mais compreensível se utilizarmos algumas estruturas sintáticas simples, como SUBSTANTIVO VERBO SUBSTANTIVO. Isso pode ser conseguido mediante o emprego de várias listas, uma para cada categoria gramatical. Seria possível, então, escolher uma palavra de cada lista de acordo com a estrutura gramatical desejada.

Ensinando sintaxe

Vamos ampliar as habilidades poéticas da tartaruga utilizando como exemplo uma estrutura gramatical do tipo VERBO SUBSTANTIVO. Começaremos definindo alguns conteúdos para VERBO e SUBSTANTIVO:

```
FACA "VERBO [COMPRAR ALUGAR VENDER]
FACA "SUBSTANTIVO [CASA MOTO SITIO
  CARRO]
```

Com estes conteúdos e o auxílio de dois novos programas, teremos um exemplo completo:

```
AP POEMA2 :ESTRUTURA
  SE :ESTRUTURA = [ ] ENTAO MOSTRE"' '
  PARE
  POEMA2.1 PRIMEIRO :ESTRUTURA
  POEMA2 SEMPRIMEIRO :ESTRUTURA
FIM
AP POEMA2.1 :PL
  SE :PL = "SUBSTANTIVO ENTAO
    MOSTRAR"' '
  MOSTRAR GERAR :SUBSTANTIVO
  SE :PL = "VERBO ENTAO MOSTRAR"' '
  MOSTRAR GERAR :VERBO
FIM
```

Note que o programa GERAR, anteriormente definido, também foi utilizado. Para se rodar o programa POEMA2, devemos chamá-lo informando a estrutura gramatical desejada:

```
POEMA2 [VERBO SUBSTANTIVO]
```

A cada execução, você obterá uma frase diferente, como: ALUGAR SITIO, COMPRAR CASA etc. O programa POEMA2.1 poderia ser melhorado com o uso do comando primitivo VALOR. Para entender melhor este comando, siga o exemplo abaixo:

```
FACA "QUALQUER "OBJETO
FACA "OBJETO "CARRO
```

Com o auxílio do comando primitivo MOSTRE, podemos observar alguns resultados interessantes:

- MOSTRE :OBJETO resulta em CARRO;
- MOSTRE :QUALQUER resulta em OBJETO; e
- MOSTRE VALOR :QUALQUER resulta em CARRO.

O comando VALOR fez com que QUALQUER fosse substituído pelo seu valor (OBJETO); e, finalmente, é mostrado o valor de OBJETO, ou seja, CARRO. Utilizando esse conceito podemos reescrever o programa POEMA2.1:

```
AP POEMA2.1 :PL
  MOSTRAR "' '
  MOSTRAR GERAR VALOR :PL
FIM
```

Como exercício, experimente criar poemas com uma estrutura gramatical mais complexa, como:

```
[ARTIGO SUBSTANTIVO ADJETIVO VERBO
  ADVERBIO PREPOSIÇAO ARTIGO
  SUBSTANTIVO ADJETIVO]
```

Para cada item faça uma lista específica com bastante criatividade. Você obterá poemas muito originais da tartaruga.

Respostas do exercício 8

A) Cálculo de potência:

```
AP POTENCIA :A :N
  SE NAO ((INT :N) = :N) ENTAO MOSTRE
  [SOMENTE NUMERO INTEIRO-EXPOENTE]
    PARE
  SE :N = 0 ENTAO SAIDA 1
  SAIDA :A * POTENCIA :A :N - 1
FIM
```

B) Converter para hexadecimal:

```
AP VER.HEXA :NUM
  SE :NUM < 10 ENTAO SAIDA :NUM
  SE :NUM = 10 ENTAO SAIDA "A
  SE :NUM = 11 ENTAO SAIDA "B
  SE :NUM = 12 ENTAO SAIDA "C
  SE :NUM = 13 ENTAO SAIDA "D
  SE :NUM = 14 ENTAO SAIDA "E
  SE :NUM = 15 ENTAO SAIDA "F
FIM
AP HEXA :NUM
  SE :NUM = 0 ENTAO PARE
  HEXA QUOC :NUM 16
  MOSTRAR VER.HEXA RESTO :NUM 16
FIM
```

C) Testar se um número é par:

```
AP PAR? :NUM
  SE ((RESTO :NUM 2) = 0) ENTAO SAIDA
    "VERD
  SAIDA "FALSO
FIM
```

D) Calcular uma área usando o método de Monte Carlo:

```
AP MC
  DESENHE SEMT FACA "IN 0
  MC1 1000 10 100
  MOSTRAR [A AREA E]
  MOSTRE :IN
FIM
AP MC1 :NUM :XNUM :YNUM
  SE :NUM = 0 ENTAO PARE
  PONTO.ALE :XNUM :YNUM
  SE DENTRO? ENTAO FACA "IN :IN - 1
  MC1 :NUM - 1 :XNUM :YNUM
FIM
AP PONTO.ALE :XNUM :YNUM
  DEFXY SORTEIE :XNUM :YNUM
FIM
AP DENTRO?
  SE CORY < CORX * CORX ENTAO SAIDA
    "VERD
  SAIDA "FALSO
FIM
```

PROGRAMAS COMPATÍVEIS

Ao tornar-se padrão industrial em sistemas operacionais, o Programa de Controle para Microprocessadores (CP/M) mudou a vida de seu criador, Gary Kildall, um dos fundadores da Digital Research.

Gary Kildall, funcionário da Intel americana, que desenvolveu o microcomputador 8080, criou a primeira versão de seu sistema CP/M (Control Program/Monitor) em 1974. Destinava-se a apoiar o compilador do PL/M, a primeira linguagem de alto nível produzida pela Intel.

No ano seguinte, Kildall acrescentou-lhe um editor (ED), um assembler (ASM) e um debugger (DDT: debugador, eliminador de erros), oferecendo o novo sistema operacional à própria Intel, que o recusou. Assim, em sociedade com Dorothy McEwan, começou a publicar revistas de informática e a vender diretamente seu Control Program/Monitor.

Intencionalmente ou não, Kildall havia criado um sistema que diminuía em muito o maior problema do microcomputador em seus primeiros anos de existência: compatibilidade. Os três computadores de maior popularidade no fim da década de 70 (PET, Apple e Tandy) possuíam sistemas operacionais incompatíveis entre si e os produtores independentes de software tinham de optar por um dos formatos.

Isso implicava reescrever por completo o software para seu aproveitamento numa máquina diferente daquela para a qual fora projetado. Mas o CP/M modificou a situação: sua larga aceitação entre os fabricantes acabou criando um "padrão" de fato. Os que haviam escolhido os processadores Intel 8080 ou Zilog Z80 para seus produtos especificaram o CP/M porque ele proporcionava uma maneira simples de manipular o acesso a tela, impressora, discos, teclado e assim por diante. E, como sua popularidade aumentasse, uma quantidade cada vez maior de software foi lançada no mercado.

A princípio, apenas alguns usuários selecionados receberam licença de usar o Programa de Controle para Microprocessadores. Em 1976, sobrecarregado de pedidos, Kildall demitiu-se do cargo de professor de Ciência da Computação numa academia naval em Monterey e fundou a Digital Research em Pacific Grove, Califórnia.

A empresa se ramifica

Enquanto crescia a aceitação do CP/M, a Digital Research voltou sua atenção para os sistemas de usuário múltiplo e criou o MP/M (Multi Program Monitor), que deveria ser compatível com o CP/M em todos os aspectos.

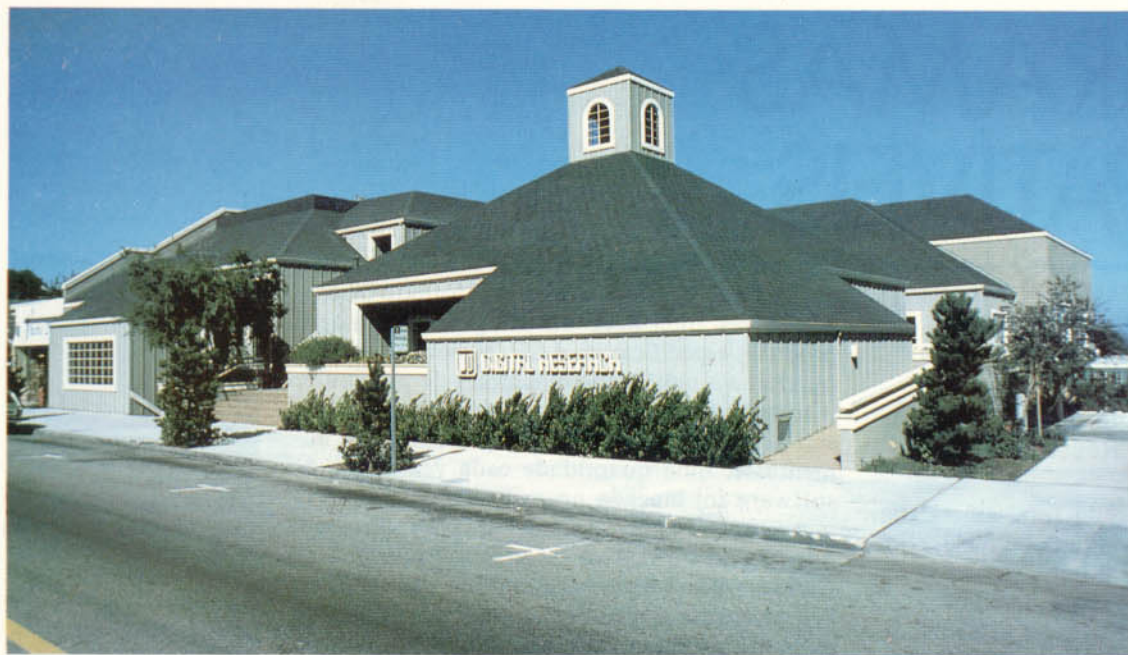
Suas primeiras versões, porém, não tiveram igual sucesso: o aumento da produção reduzira os custos dos microprocessadores para o consumidor final e a partilha de um deles entre diversos usuários deixou de fazer sentido econômico. Por esse motivo, mesmo revisto, o MP/M não se tornou popular.

Depois de levantar fundos junto a diversas empresas financeiras em 1981, a Digital Research tornou-se uma verdadeira multinacional, com forte presença na Europa, especialmente na Inglaterra, Alemanha e França.

Projetos LOGO
A Digital Research entrou no campo das linguagens com seu DR LOGO. Como outros, este tem nos gráficos um de seus pontos fortes.

Gráficos comerciais
GSX é um pacote de software gráfico pioneiro, projetado para transferir gráficos entre as diferentes máquinas e aplicações.

John Rowley, presidente da Digital Research

Gary Kildall

Sede da Digital Research em Massachusetts, EUA

Ao mesmo tempo, constituía uma das empresas com maior possibilidade de ganhar o contrato para desenvolver um sistema operacional para o recém-criado microcomputador da International Business Machines, o IBM PC.

A encomenda acabou indo para a Microsoft; apesar disso, a Digital Research não parou de crescer. Logo atualizou o CP/M para os processadores Intel 8088/8086, de forma a fazê-los muito parecidos com o MS-DOS, e deu também um passo adiante com o CP/M Concurrent.

O CP/M Concurrent é o inverso do MP/M, permitindo que diversos programas diferentes sejam executados simultaneamente. Com esse programa, um usuário pode trabalhar com três coisas diferentes ao mesmo tempo — por exemplo, planilha eletrônica, geração de relatório e editor de texto —, passando de uma para outra à vontade.

As versões existentes do CP/M Concurrent podem exibir cada uma das telas — ou apenas parte delas — simultaneamente, usando "janelas"; os novos modelos em desenvolvimento prometem executar diretamente a maioria dos programas escritos para o IBM PC-DOS.

O "microware"

Entre as decisões estratégicas que a Digital Research e muitas outras empresas de sistemas tomaram está a de passar todo o seu trabalho de desenvolvimento para a linguagem C, particularmente notável pela portabilidade. O código escrito em C precisa apenas ser recompilado para se tornar útil em outro processador, embora seus adversários argumentem que é melhor fazer um trabalho apropriado em ASSEMBLER para cada processador isoladamente. Contudo, sua popularidade vem crescendo, e uma vez que o sistema operacional UNIX, amplamente utilizado, é escrito em C, a tendência a usar esta linguagem parece ser irreversível.

A Digital Research tem sido coerente em seu ponto de vista de que a verdadeira portabilidade só é possível por meio das linguagens de alto nível, das quais fornece grande variedade. Na faixa de micros, desenvolveu um Personal BASIC, um Personal CP/M e sua própria versão de LOGO. O Personal CP/M, como o CP/M-86, é projetado para ser armazenado em ROM. A Digital descreve isso como sendo "microware".

VIP e GSX

Em fins de 1984, a empresa de Gary Kildall dedicava-se aos projetos do VIP e do GSX. O VIP é um gerador de telas padronizadas que permite a apresentação de uma interface uniforme ao usuário, independentemente do pacote de aplicações que estiver sendo executado.

Várias aplicações podem usar as mesmas informações. Nesse aspecto, o VIP é semelhante à tecnologia do Lisa e do Macintosh da Apple, só que exige bem menos memória. O VIP pode ser executado em qualquer computador com mais de 50 Kbytes de RAM e equipado com espaço de disco de 150 Kbytes ou mais.

O GSX faz pelos gráficos o que o CP/M faz pelos discos. Usa um conjunto padrão de funções gráficas, destinadas a várias peças diferentes de hardware.

Um programa GSX pode ser executado numa tela em cores ou em preto e branco, numa impressora matricial e num plotter, sem qualquer modificação. Além da relativa falta de software, havia também dificuldades na criação de novos padrões de gráficos, pois o sistema não produz a mesma qualidade dos programas escritos para uma única máquina.

Depois de lançar produtos como GSX, VIP e LOGO, e com potencial para concorrer com a Microsoft no campo de softwares aplicativos, a Digital Research coloca-se como uma das maiores firmas internacionais do ramo.

SUPERANDO LIMITAÇÕES

Para os deficientes, o micro é bem mais do que uma possibilidade de avanço no aprendizado. Representa, com freqüência, o único meio de superar barreiras entre eles e seu ambiente.

Muitos adultos e crianças necessitam de métodos especiais na educação. Isso devido tanto a problemas físicos, como surdez ou cegueira, quanto a retardo mental. Cada deficiência apresenta problemas específicos, resultantes, muitas vezes, mais do preconceito ou da insensibilidade social do que da deficiência em si.

As escolas especiais são poucas e mal servidas de verbas governamentais. Conseqüentemente, esse não é um mercado lucrativo para os projetistas de hardware e software — desvantagem superada pela dedicação dos voluntários que trabalham nessa área. Um grupo inglês conseguiu angariar o equivalente a quase 20.000 dólares para desenvolver um telefone especial e de baixo custo para surdos. Em muitos países, professores, programadores e engenheiros tomam a iniciativa de desenvolver recursos eletrônicos para o ensino especial.

Na Inglaterra, Centros de Recursos de Microeletrônica para a Educação Especial (Special Education Microeletronics Resources Centres, SEMERC) foram estabelecidos em Manchester, Newcastle e Redbridge para atender a essas necessidades. Uma de suas principais funções — além de avaliar e implantar sistemas — consiste em preparar professores para deficientes e mantê-los informados sobre novidades.

Em alguns casos, é a comunicação que apresenta os maiores problemas. Uma pessoa lúcida e inteligente pode estar fisicamente incapacitada de se comunicar com o mundo exterior. A microtecnologia vem criando canais de comunicação, como o Photonic Wand (''bastão fotônico''), um sensor óptico montado num capacete de plástico e controlado por movimentos da cabeça. Ele torna possível a pessoas com deficiência na fala e no controle motor operar um micro. O bastão conecta-se ao micro pelo canal de entrada analógica. O sensor óptico é semelhante a uma caneta óptica e move um cursor pela tela em resposta a movimentos da cabeça.

Um programa processador de texto chamado Write pode ser usado com o bastão. Ele apresenta o alfabeto na tela, e as letras, maiúsculas ou minúsculas, são escolhidas apontando-se os caracteres. Há uma função de edição e listas de palavras disponíveis. O texto pode ser impresso, gravado ou apagado.

Outro programa, o Paint, permite desenhar com seis cores, e um terceiro, denominado Music, apresenta na tela um teclado em que as notas são selecionadas com o cursor e tocadas.

O Photonic Wand pode ser usado por pessoas com movimentos instáveis das mãos e mesmo por crianças de apenas oito anos. O SEMERC de Manchester desenvolveu um trabalho com a finalidade de conectar o sensor óptico a um sintetizador de fala. As respostas padronizadas em conversas telefônicas, do tipo ''Repita, por favor'', são selecionadas na tela e então faladas pelo sintetizador. Espera-se ampliar o vocabulário, assim como desenvolver um método simples de conectar o sintetizador ao telefone.

Os computadores são projetados para quem tem a capacidade da visão. Monitores, impres-

Trabalho pioneiro
Um terminal de trabalho inglês combina uma série de adaptações úteis às pessoas com diferentes deficiências. O Perkins Brailler possibilita a pessoas cegas fornecer informações ao computador e o sintetizador de voz propicia um retorno sonoro. A apresentação do texto na tela foi ampliada para beneficiar os portadores de visão parcial. O terminal de trabalho também se conecta a um teclado conceitual. Embora a foto mostre uma impressora comum, é possível (a preço elevado) a aquisição de impressora especial em Braille.

soras e plotters são saídas visuais e todos os programas utilizam a tela para apresentar seus dados. A Open University (Universidade Aberta) britânica tem buscado formas de adaptar o hardware disponível e de baixo custo para ser usado por cegos, que colaboram no projeto. Terminais de trabalho, consistindo em um microcomputador BBC, unidade de disco, monitor, impressora, sintetizador de voz, teclado conceitual e um Perkins Brailler adaptado foram instalados em escolas e em outros locais.

O Perkins Brailler foi inventado na década de 40 para datilografia em Braille (código de pontos gravados em relevo no papel, para possibilitar a leitura pelo toque). Ligado ao computador, permite ao cego a leitura da saída impressa. Elaborou-se também um software por meio do qual o computador converte o Braille em texto normal; este pode então ser armazenado, editado ou impresso.

Vários processadores de texto que sintetizam a voz foram desenvolvidos para uso com o teclado padrão. Caracteres digitados e teclas especiais são confirmados por um som. A tecla [Delete] também fala o nome do caractere que foi apagado. O texto é editado utilizando-se um cursor comandado pela voz. Enquanto ele se move na tela, os caracteres, as palavras ou as sentenças são falados. O movimento do cursor pode ser interrompido para adicionar ou apagar partes do texto. Os programas mais sofisticados têm recursos para formatar o texto antes de imprimi-lo. Podem-se definir margens, títulos e espaços para produzir uma cópia final bem-acabada.

A grande vantagem do teclado conceitual sobre o teclado padrão é que ele se adapta a diversas aplicações; pode, por exemplo, ser dividido em quatro seções; uma figura é desenhada em cada quarto, possibilitando o controle de uma tartaruga eletrônica do tipo utilizado pela linguagem LOGO. A pressão em cada uma das diferentes seções envia a tartaruga para a frente, para trás, para a esquerda ou para a direita, o que — no caso de crianças sem capacidade física — é mais simples do que digitar uma instrução. Já foram desenvolvidos um software e uma linguagem, a Starset, para o teclado conceitual.

O mouse usado com o computador Apple Macintosh também evita a "barreira do teclado" e tem amplo potencial no ensino especializado.

Há modernos interruptores que podem ser operados por meios não manuais. Num deles, projetado para pessoas sem movimentos voluntários, dois pequenos discos de metal colocados na pele, próximo aos olhos, detectam o movimento horizontal do globo ocular, e os sinais elétricos decorrentes são amplificados para controlar os interruptores.

No Departamento de Ciência e Sociedade da Universidade de Bradford, uma luz laranja de brilho intermitente foi conectada a uma tartaruga eletrônica para que um jovem deficiente visual pudesse enxergá-la. Usando um teclado conceitual, ele pôde movimentar a tartaruga pelo chão e ver o resultado de suas ações. Para esse rapaz e para muitas crianças deficientes, experiências desse tipo representam um grande passo no sentido de levá-los a participar de seu ambiente, deixando de ser observadores passivos.

Uma especialista inglesa, a dra. Sylvia Weir, apresentou em 1984, numa conferência, notáveis exemplos de progressos alcançados por crianças deficientes físicas e mentais graças ao uso do LOGO e das tartarugas. Um menino autista, de se-

O toque mágico
O Photonic Wand propicia, mesmo a pessoas altamente deficientes, impossibilitadas de falar ou de mover os membros, o controle de um micro. O movimento do bastão é detectado por um sensor óptico e traduzido para um sinal analógico processado pelo computador.

te anos e meio, pôde controlar uma tartaruga por meio de uma caixa com botões, dispositivo semelhante ao teclado conceitual. Ficou tão entusiasmado com a experiência que se pôs a falar pela primeira vez.

A dra. Weir também vem usando o LOGO numa escola especial, com um grupo de adolescentes mentalmente alerta, mas que sofrem de paralisia cerebral. Controlar o computador os ajuda a superar a passividade a eles imposta por sua deficiência.

Um esforço conjunto do SEMERC de Manchester e um grupo local de computação produziu o Micromike, dispositivo para auxiliar crianças com deficiência de fala. Um microfone adaptado, que se conecta ao BBC Micro, possibilita a crianças controlarem várias atividades na tela usando a voz. O software desenvolvido para isso inclui o City, programa com o qual se pode desenhar a silhueta dos edifícios de uma cidade. A altura e a largura dos prédios são determinadas pelo volume e duração da voz. Outros programas permitem o movimento de estrelas no céu, a pilotagem de um barco por corredeiras e o salvamento de um canoeiro por um helicóptero. Eles dão às crianças a oportunidade de aperfeiçoar o controle da voz.

Do mesmo modo que o ensino normal, a educação especial se encontra defasada em relação às conquistas tecnológicas, voltadas mais para áreas administrativas e industriais. A escassez de verbas e de conhecimento especializado retarda o surgimento de hardware e de software para deficientes, e as escolas só lentamente vão adotando a tecnologia da computação. Apesar de tudo, um trabalho fascinante já começa a ser desenvolvido.

Especialmente para tetraplégicos — paralíticos dos quatro membros, que em alguns casos só podem mover olhos e boca —, a Empresa Brasileira de Microinformática (Embramic), de Porto Alegre, instalou um sistema que lhes permite, por meio de sopros apenas, a execução de tarefas antes impossíveis.

Trata-se do Embramic 2000, que reúne um micro Maxxi com unidades de disco, dispositivos eletromecânicos para acionamento e software aplicativo. Soprando num tubo próximo à boca, o deficiente liga o computador e escolhe no monitor de vídeo — em que se desloca um cursor luminoso — a opção desejada: acender luzes, ligar o televisor, abrir portas e discar o telefone. Até princípios de 1985, o Embramic 2000 era o único no gênero no Brasil. Produtos semelhantes, nos Estados Unidos, eram mais onerosos e comandados por voz.

Teclado conceitual
Algumas crianças deficientes não possuem capacidade física para usar o teclado padrão. Um teclado liso e sensível ao toque foi desenvolvido para elas. Chamado de "teclado conceitual", possibilita que películas de acetato substituíveis sejam colocadas sobre ele, adaptando-o a várias aplicações.

E os superdotados?

Crianças superdotadas também têm necessidades especiais de educação e o uso dos micros fornece uma alternativa estimulante para elas, assim como para alguns tipos de deficientes. Muitas crianças com descontrole motor ou com dislexia (dificuldade de ler) são intelectualmente superdotadas.

Nesses casos, o microcomputador oferece um meio de obter controle sobre o ambiente e superar os obstáculos ao desenvolvimento individual, proporcionando à criança superdotada:

- Oportunidade de desenvolver capacidades e interesses em seu próprio ritmo.
- Novos métodos criativos de abordagem de problemas.
- Um meio de comunicação e cooperação com outros estudantes superdotados.
- A oportunidade de praticar exercícios monótonos e cansativos de um modo que mantenha seu interesse.
- Um novo campo para descobertas.

ARQUIVOS NA RAM

Os métodos de manipulação de arquivos tendem a ser específicos para cada tipo de máquina. Como aproveitá-los em micros que só utilizam fitas cassete?

Micros diferentes exigem técnicas distintas para a manipulação de arquivos. Isso torna indispensáveis a adaptação dos programas para cada equipamento e o conhecimento de como a configuração e os comandos próprios de seu sistema se relacionam com os princípios gerais.

Quanto ao armazenamento de arquivos de dados num micro padrão que só utilize cassete, há um primeiro ponto a observar: sistemas desse tipo, por sua própria natureza, não podem lidar com arquivos de acesso aleatório, e os dados são acessados na ordem em que foram armazenados — isto é, seqüencialmente.

A utilização de arquivos seqüenciais envolve a leitura de informações de um arquivo, a manipulação dessas informações e a posterior armazenagem dos dados modificados num segundo arquivo. Assim, devemos ter dois arquivos em uso ("abertos") ao mesmo tempo. Como um gravador cassete não permite passar de uma posição a outra da fita de modo rápido e preciso, esse sistema de "dois arquivos" não é o mais adequado. Constituem exceções alguns sistemas que dispõem de duas entradas para cassete — uma para leitura e outra para gravação.

Por isso, a maioria dos equipamentos disponíveis no nosso mercado só pode acessar um único arquivo seqüencial, o que impõe, na prática, algumas limitações. O arquivo deve ser lido na memória, antes de ser usado; então, se houver modificações a fazer, deve ser novamente gravado no cassete. Procede-se por intervalos, na execução do programa, ou de uma só vez, no seu final. Os arquivos de dados devem ser pequenos o suficiente para caber no espaço da RAM, ainda não utilizado pelo próprio programa. A maior parte dos micros está, dessa forma, restrita a arquivos pouco extensos.

Três métodos principais foram desenvolvidos para armazenagem de informações em cassete. O mais simples não utiliza arquivos de dados separados; ao invés disso, todas as variáveis em uso são armazenadas juntamente com o programa, sempre que o comando SAVE for empregado. Esse método é utilizado pelo TK 82 e também pelo CP 200, da linha Sinclair. Quando se necessita de outro arquivo de dados, nova cópia do programa é usada e gravada com esses dados.

Se tal versão for carregada na memória, os dados serão automaticamente lidos e colocados de volta em suas respectivas variáveis. A vantagem desse método é sua simplicidade — o usuário precisa apenas assegurar-se de que o programa completo seja bem gravado e carregado.

Um sistema mais sofisticado requer um BASIC capaz de armazenar e recuperar matrizes específicas. No Oric Atmos, por exemplo, o comando STORE A$, "NOME" armazena o string A$ na fita, enquanto RECALL A$, "NOME" serve para recuperá-lo. O string inteiro (A$(1), A$(2) etc.) é gravado, apesar de os comandos STORE e RECALL não especificarem seu tamanho. Isso é feito automaticamente quando se dimensiona a matriz no começo do programa.

Esse sistema exige o controle do número de registros da matriz utilizados em seu programa. Uma solução é armazenar a contagem dos registros na matriz antes que ela seja gravada. A maioria dos equipamentos permite um índice zero, de forma que um elemento do tipo A$(0,0) pode ser usado como contador de registros.

O contador de registros será uma variável numérica (em nosso programa usaremos R), mas, se uma matriz de strings estiver sendo usada, ele deverá converter-se numa variável string mediante a seguinte instrução: A$(0,0) = STR$(R). Uma vez recarregada a matriz no computador, reinicializa-se R com: R = VAL(A$(0,0)).

Como muitos micros não possibilitam procedimentos sofisticados de manipulação de arquivos, estes devem ser simulados. Com o arquivo gravado na memória, é fácil usar matrizes em BASIC para tratá-lo como se fosse um arquivo de acesso aleatório.

Vamos supor que uma matriz bidimensional de strings seja usada para armazenar os dados, com uma instrução como DIM A$(100,3). O primeiro índice na matriz designa um registro determinado e o segundo indica um entre quatro campos. Desse modo, os dados são armazenados no formato de tabela — o que equivale a um arquivo de acesso aleatório.

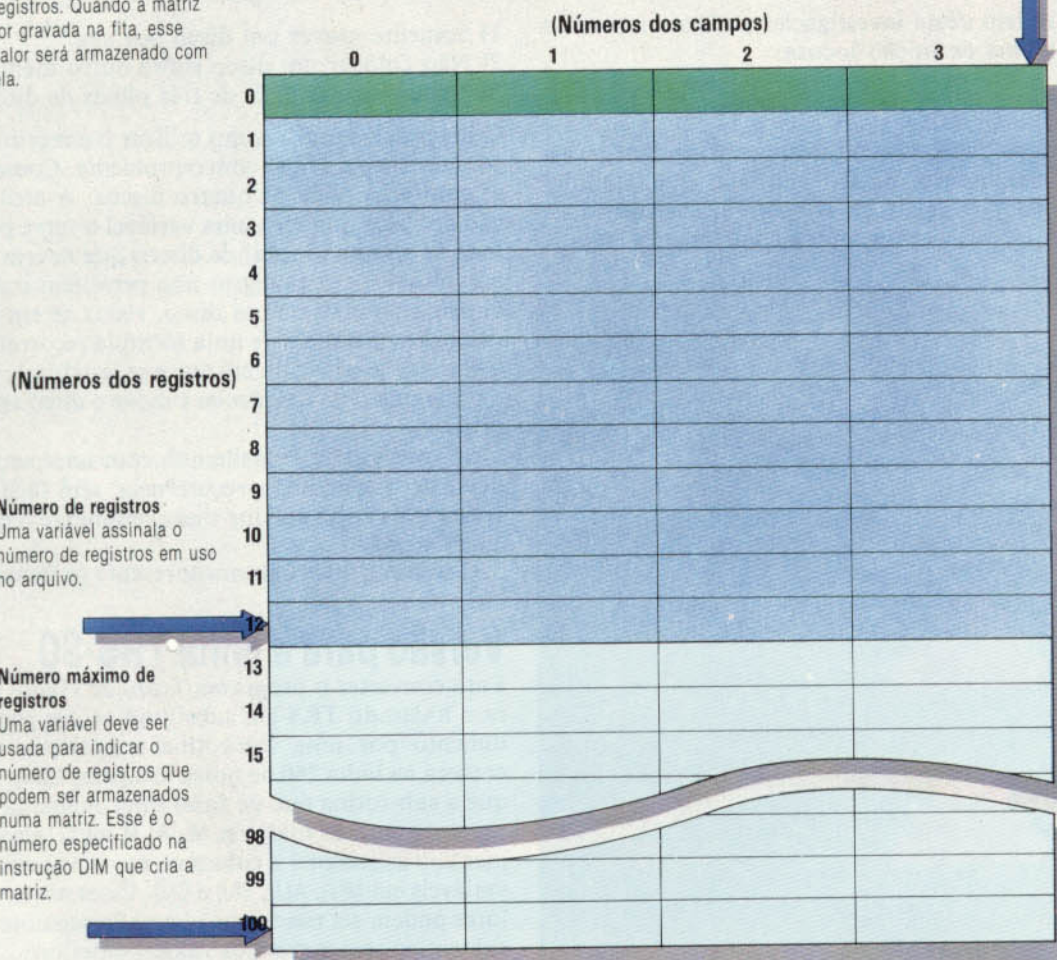
Outro recurso encontrado em algumas máquinas é o comando APPEND. Ele permite acrescentar dados no final de um arquivo seqüencial sem que seja necessário lê-lo todo para criar uma nova versão. Certos computadores possuem um comando para se pular determinado número de registros de um arquivo seqüencial e, desse modo, simular um arquivo de acesso aleatório.

Embora a manipulação de arquivos dependa muito do tipo de equipamento, os princípios gerais são sempre os mesmos, não importando o micro utilizado.

Como armazenar registros e campos numa matriz em BASIC

Uma matriz bidimensional de strings pode ser usada para simular um arquivo de acesso aleatório. O primeiro índice identifica um registro específico e o segundo, os campos dentro do registro.

Número de campos
É bastante útil ter esse número definido numa variável no começo do programa. Isso facilita a mudança posterior para registros maiores, porque significa alterar apenas uma instrução.

Registro de identificação
A$(0,0) é usado para armazenar o número de registros. Quando a matriz for gravada na fita, esse valor será armazenado com ela.

Número de registros
Uma variável assinala o número de registros em uso no arquivo.

Número máximo de registros
Uma variável deve ser usada para indicar o número de registros que podem ser armazenados numa matriz. Esse é o número especificado na instrução DIM que cria a matriz.

Eliminação de um registro
Este programa remove da matriz o registro número N. Todos os registros abaixo deste são movidos para uma posição mais acima, anulando, assim, os dados anteriormente armazenados na posição N.

```
100 LET R=R + 1
110 IF > M THEN PRINT "MATRIZ
CHEIA":RETURN
120 FOR I=R TO N + STEP - 1
130 FOR J=0 TO F
140 LET A$(I,J)=A$(I - 1,J)
150 NEXT J: NEXT I
170 LET A$(N,0)=N$: LET A$(N,1)=C$
180 LET A$(N,2)=D$: LET A$(N,3)=E$
190 RETURN
```

Inserção de um registro
Este programa acrescenta um novo registro na matriz, na posição N. Todos os registros além do ponto da inserção são deslocados para uma posição mais abaixo, criando assim o espaço necessário para os novos dados.

```
200 FOR I=N TO R - 1
210 FOR J=0 TO F
220 LET A$(I,J)=A$(I + 1,J)
230 NEXT J: NEXT I
240 LET R=R - 1
250 RETURN
```

TORRE DE HANÓI

O conhecimento da recorrência, técnica usada na programação avançada, pode acrescentar novas dimensões a programas em BASIC. O quebra-cabeça Torre de Hanói, aqui desenvolvido, é um exemplo.

O objeto desta investigação pode ser resumido por uma definição jocosa:

Recorrência: *ver* Recorrência

Essa definição circular demonstra um aspecto essencial da recorrência — ou seja, algo que é definido por meio de si mesmo. Deixou-se de considerar, porém, outra característica importante: para uma recorrência (ou recursion) ser operável, deve haver uma saída da circularidade.

O quebra-cabeça Torre de Hanói, que ilustra a recorrência, consiste numa pilha de discos dispostos por ordem de tamanho: o disco maior fica embaixo da pilha e o menor em cima. Para resolver o quebra-cabeça, é necessário passar todos os discos da primeira pilha para a segunda, de acordo com as regras:

1) Somente mover um disco por vez.
2) Não colocar um disco sobre outro menor.
3) Nunca formar mais de três pilhas de discos.

O diagrama mostra como utilizar o conceito de recorrência para lidar com o problema. Começa-se com uma pilha de quatro discos. A atribuição do valor quatro a uma variável n serve para indicar o número total de discos que devem ser movidos. Como as regras não permitem o movimento de mais de um disco, reduz-se em 1 o valor de n por meio de uma fórmula recorrente. Continua-se o cálculo até que n se iguale a 1. Aí, o programa pára de calcular e move o disco apropriado.

Se você estiver trabalhando com uma versão de BASIC que admite a recorrência, será fácil escrever um programa que siga exatamente o processo acima.

O restante do programa apresenta as imagens na tela.

Mova um disco

O programa pede ao jogador que digite o número de discos que serão movidos. Esse número é colocado na pilha A. Em seguida, o algoritmo recorrente move um disco por vez, formando as pilhas B e C, nos três primeiros estágios. Por fim, a pilha inicial de n discos é reduzida a zero: não há mais discos para mover, como mostra o resultado final. Ao término dos movimentos, n discos estarão reposicionados em ordem seqüencial na pilha B.

Versão para a linha TRS-80

Para converter o programa Torre de Hanói para o BASIC do TRS-80, substituímos um procedimento por uma sub-rotina recorrente, que começa na linha 260 de nossa listagem. Cada vez que a sub-rotina tem de fazer uma chamada recorrente para as matrizes M, A, B ou C, ela aumenta o contador J e coloca os novos valores de variáveis em M(J), A(J), B(J) e C(J). Esses novos valores podem ser usados na chamada seguinte da sub-rotina sem mexer nos valores anteriores. No final da sub-rotina, o valor J é reduzido, restaurando assim os antigos valores. Esse método sempre pode ser usado para escrever sub-rotinas recorrentes em BASIC, por mais complexa que seja a recorrência.

Problemas recorrentes
Assim aparece o jogo Torre de Hanói ao ser rodado no micro Spectrum, da Sinclair. Se você tentar acompanhar a resolução do problema por este micro, preste muita atenção, pois ele é bem rápido!

Linha TRS-80

```
100 REM***********************************
120 REM*          PARA LINHA TRS-80        *
130 REM*           TORRE DE  HANOI         *
150 REM***********************************
170 CLS: CLEAR 1000: DIM M(100), A(100), B(100), C(100)
180 DIM D$(10), H(3), P(3,10)
190 GOSUB 690
200 PRINT@0,"";:INPUT"QUANTOS DISCOS ";N
210 IF N<1 OR N>10 THEN GOTO 200
220 GOSUB 620
230 J=1: M(J)=N: A(J)=1: B(J)=2: C(J)=3
240 GOSUB 260
250 GOTO 200
260 IF M(J)=1 THEN GOSUB 450   :RETURN
270 J=J+1
280 M(J)=M(J-1)-1
290 A(J)=A(J-1)
300 B(J)=C(J-1)
310 C(J)=B(J-1)
320 GOSUB 260
330 M(J)=1
340 A(J)=A(J-1)
350 B(J)=B(J-1)
360 C(J)=C(J-1)
370 GOSUB 260
380 M(J)=M(J-1)-1
390 A(J)=C(J-1)
400 B(J)=B(J-1)
410 C(J)=A(J-1)
420 GOSUB 260
430 J=J-1
440 RETURN
450 PA=A(J): PB=B(J)
460 M$=D$(P(PA,N+1-H(PA)))
470 FOR I=14-H(PA) TO 2 STEP -1
480 PRINT@(I-1)*64+15*(PA-1),M$;
490 PRINT@I*64+15*(PA-1),B$;
500 NEXTI
510 FOR I=15*(PA-1) TO 15*(PB-1) STEP SGN(PB-PA)
520 PRINT@1*64+I,M$;
530 PRINT@1*64+I,B$;
540 NEXTI
550 FOR I=1 TO 12-H(PB)
560 PRINT@I*64+15*(PB-1),B$;
570 PRINT@(I+1)*64+15*(PB-1),M$;
580 NEXTI
590 H(PB)=H(PB)+1: P(PB,N+1-H(PB))=P(PA,N+1-H(PA))
600 P(PA,N+1-H(PA))=0: H(PA)=H(PA)-1
610 RETURN
620 CLS
630 FORI=1 TO N
640 PRINT@(13-N+I)*64,D$(I);
650 P(1,I)=I: P(2,I)=0: P(3,I)=0
660 NEXTI
670 H(1)=N: H(2)=0: H(3)=0
680 RETURN
690 FORI=1TO9STEP2
700 D$(I)=STRING$(5-INT(I/2)," ")+CHR$(170)+STRING$(2*INT(I/2)
   ,191)+CHR$(149)+STRING$(5-INT(I/2)," ")
710 D$(I+1)=STRING$(5-INT(I/2)," ")+STRING$(I+1,191)+STRING$
   (5-INT(I/2)," ")
720 NEXTI
730 B$=STRING$(12," ")
740 RETURN
```

SUPERSECRETO

Os programas podem ser construídos a partir de blocos independentes chamados "módulos". Veremos, a seguir, como utilizá-los no desenvolvimento de um programa completo.

Quando se constrói um programa, é boa idéia desenvolver uma estrutura genérica. Nesta, as rotinas e aplicação geral podem ser utilizadas por outras, de especialização crescente, em níveis mais altos — todas sob a direção de um único módulo de controle no topo. Essa estrutura piramidal permite-nos usar um método de programação chamado "refinamento" ou "projeto topo-base".

O primeiro passo é a elaboração do programa de controle, localizado no topo. Descrevemos suas funções em termos de chamadas a rotinas de nível "inferior"; em seguida, passamos para o nível de baixo e descrevemos o trabalho de cada uma das rotinas chamadas pelo módulo do topo. Cada rotina é descrita em termos das rotinas que ela deve chamar; o processo é repetido até que se alcance o nível mais baixo. Nesse estágio, as funções desempenhadas pela rotina que estamos descrevendo são tão simples que podem ser definidas com o uso da própria linguagem de programação.

Como exemplo, vejamos o projeto de um jogo de forca. Em vez de o jogador adivinhar uma palavra escolhida pelo programa, como na maioria das versões para computador desse jogo, queremos um programa que adivinhe uma palavra escolhida por nós. Um modo de conseguir isso sem dar ao programa uma longa lista de palavras é fornecer dados sobre a probabilidade da ocorrência de determinadas seqüências de letras.

```
100 REM INICIALIZAR VARIAVEIS E MATRIZES
500 REM ****** ROTINA DE CONTROLE ******
510 REM
520 GOSUB 1000 : REM TITULO E TELAS DE
    AJUDA
530 GOSUB 2000 : REM MONTA A TELA
540 GOSUB 4000 : REM LE O TAMANHO DA
    PALAVRA
550 GOSUB 8000 : REM CARREGA O ARQUIVO
    DE PALAVRAS APROPRIADO
560 GOSUB 3000 : TENTA UMA LETRA
570 GOSUB 4500 : REM VERIFICA SE A
    TENTATIVA FOI CORRETA
580 GOSUB 5000 : REM ATUALIZA A TELA
590 IF JOGO_NAO_TERMINADO THEN 560 :
    REM CONTINUA FAZENDO TENTATIVAS ATE
    O FINAL DO JOGO
600 IF CORRETO THEN GOSUB 10000 ELSE
    GOSUB 11000 : REM FINALIZA DE ACORDO
    COM O RESULTADO
610 GOSUB 6000 : REM PERGUNTA SE O
    JOGADOR QUER JOGAR DE NOVO
620 IF SIM THEN 530 : REM INICIA DE NOVO
630 GOSUB 7000 : REM TCHAU E FIM
640 END
```

Sabe-se de antemão que algumas coisas devem ser feitas: é preciso definir variáveis, dimensionar matrizes, diagramar e atualizar a tela do jogo. Além disso, é necessário escrever rotinas que contem pontos, sugiram letras e, por fim, encerrem o jogo.

Vamos partir de uma instrução REM simples para indicar que será preciso definir variáveis e matrizes — deixando para completar os detalhes mais tarde. A rotina de controle propriamente dita é apenas um par de loops. O loop externo (linha 620) verifica se o jogador quer finalizar a sessão, enquanto o loop interno (linha 590) testa se o jogo terminou.

Para testar a rotina de controle, criam-se sub-rotinas simuladas correspondentes aos GOSUBs. Cada GOSUB deve ter uma linha REM para explicar sua função e deve começar numa linha com número conveniente — de preferência um número "redondo", como 1000 ou 5000. Rotinas com funções similares devem ter seus números de linhas padronizados; isso vai facilitar muito quando as rotinas forem transportadas de um programa para outro. Por exemplo, instruções de jogo podem estar sempre contidas numa sub-rotina que comece na linha 1000.

A rotina de controle é curta e simples. Cabe inteira na tela: isso torna mais fácil eliminar eventuais problemas, em comparação com programas que se estendam por várias telas. As variáveis JOGO NAO TERMINADO, CORRETO e SIM são sinalizações (flags) colocadas nas várias sub-rotinas chamadas pela rotina de controle; são usadas para determinar se o programa de controle funciona corretamente. É muito fácil detectar qualquer erro de lógica nessa rotina de controle tão simples.

Nesse ponto, deve-se examinar a estrutura do programa, para ver se este se comporta da maneira esperada em todas as circunstâncias. Pode-se, também, melhorar o projeto do programa. Deixar, por exemplo, as instruções acessíveis em qualquer estágio do jogo, manter um registro de quantos jogos o computador ou o jogador ganharam, ou, ainda, listar as palavras que derrotaram o computador. Todas essas modificações podem ser feitas nesse estágio.

A etapa seguinte consiste na especificação de cada uma das rotinas chamadas pelo programa de controle. Nossas listagens sugerem duas des-

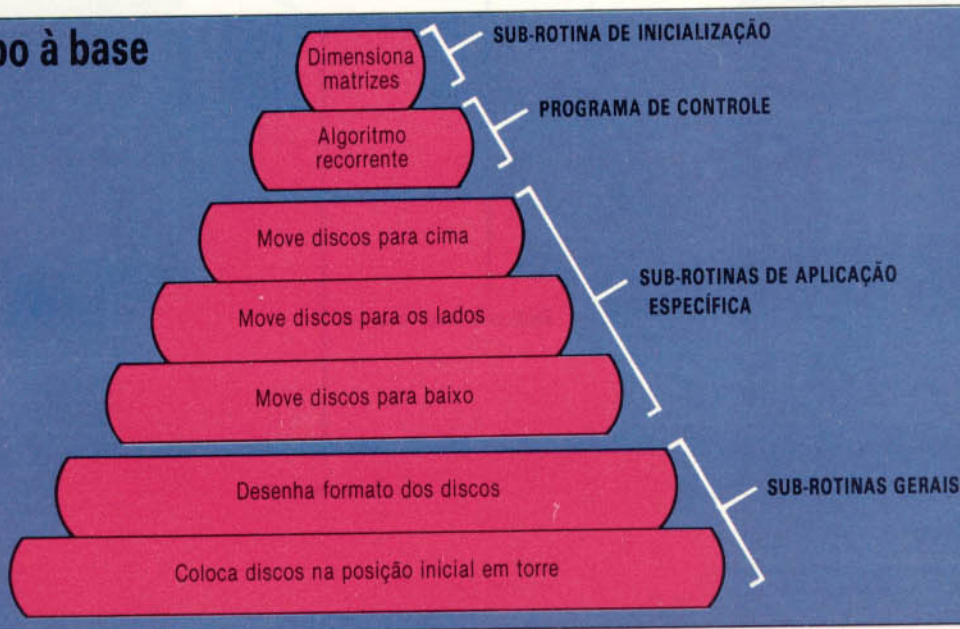
sas rotinas. A primeira, que começa na linha 4000, pede ao usuário um número entre 1 e 20 (o comprimento da palavra). Ela usa uma sub-rotina de aplicação geral que se presume existir na linha 51000, a qual toma uma variável alfanumérica (string) especificada em PROMPT$, a imprime e aceita um número inteiro ao qual o usuário dá entrada. Se esse número não estiver entre os limites definidos por MIN% e MAX%, surgirá uma mensagem de erro e será pedido ao usuário que entre com um novo número. Essa sub-rotina pode facilmente ser usada em outros programas; recomenda-se que o usuário monte uma biblioteca desses módulos genéricos.

```
4000 REM PEDE AO JOGADOR O TAMANHO DA
     PALAVRA
4010 REM
4020 PROMPT$ = "QUANTAS LETRAS TEM A SUA
     PALAVRA ?"
4030 MIN% = 1
4040 MAX% = 20
4050 GOSUB 51000 : REM ACEITA UM NUMERO
     INTEIRO ENTRE MIN% E MAX%
4060 COMPR_L% = RESP% : REM RESP% E A
     RESPOSTA USADA PELA SUBROTINA NA
     LINHA 51000
4070 RETURN
8000 REM CARREGA ARQUIVO DE PALAVRAS
8010 REM
8020 IF COMPR_L% > 7 THEN ARQ_L% = 8
     ELSE ARQ_L% = COMPR_L%
8030 ARQNUM_L$ = STR$(ARQ_L%)
8040 ARQNOME$ = "TABELA" + ARQNUM_L$
8050 GOSUB 9000 : REM OPEN, READ E CLOSE
     O ARQUIVO DE PALAVRAS COM O COMPRI-
     MENTO APROPRIADO
8060 RETURN
```

A outra rotina, que começa na linha 8000, usa variáveis locais (ARQ L% e ARQNUM L$). Assumimos que os dados necessários para se adivinhar uma letra estão, por exemplo, em oito conjuntos de tabelas. Como só queremos um conjunto de dados na RAM de cada vez, devemos construir um string ARQNOME$ para guardar o nome do arquivo de dados e então chamar a sub-rotina na linha 9000 para que o leia.

Em muitos casos, percebe-se que o programa se move diretamente de uma rotina para outra. No entanto, pode-se criar uma rotina extra para chamar sucessivamente cada uma das outras. Isso pode parecer desnecessário, mas permite que se controle o fluxo do programa e apresenta ainda a vantagem de manter os módulos separados, de modo que podem ser acrescentados a outros programas.

Um recurso para facilitar o projeto de sub-rotinas "versáteis", transportáveis de um programa para outro, é a substituição de constantes por variáveis. É também importante que todas as sub-rotinas sejam bem documentadas. A documentação deve especificar a aplicação exata da rotina, dando detalhes das variáveis usadas, dos valores esperados para a entrada e a saída e de quaisquer efeitos secundários — deslocamentos do cursor, mudança do mapa da memória, fechamento de arquivo etc.

Uma codificação padronizada também facilita bastante; certifique-se de que todos os números de linha obedecem a um intervalo fixo, de que os títulos e comentários estão restritos a um dado número de linhas no começo da rotina e de que o RETURN está sempre na última linha. Não esqueça de anotar o número da primeira e da última linha de cada rotina. Quando for necessária uma rotina de sua biblioteca, confirme se o programa tem um espaço apropriado em seus números de linha e então inclua a sub-rotina no programa com um comando MERGE. Se o seu micro não tiver o comando MERGE, poderá ser usado um editor de texto para combinar programas gravados no formato ASCII em vez da forma "simbólica" usual. Se isso não for possível, as sub-rotinas da sua biblioteca deverão ser digitadas cada vez que forem usadas. Contudo, o fato de não precisarem ser elaboradas já justifica o trabalho extra.

Programação topo-base
O diagrama ilustra o princípio da programação topo-base. É o programa da Torre de Hanói, mostrado anteriormente. O nível mais alto da estrutura representa o programa de definição, que deve ser completado antes de se executar o resto do programa. O programa de controle, no diagrama, representa o algoritmo recorrente, que efetua os cálculos e, se preciso, chama as outras sub-rotinas. As sub-rotinas de aplicação específica são usadas para movimentar os discos de uma pilha para outra na tela. As duas últimas seções do diagrama, sub-rotinas gerais, representam as duas seções finais do programa e são usadas para formatar a tela inicial e criar o desenho dos discos. Compare esse diagrama com a listagem e veja que o programa possui a mesma estrutura.

ALFA 3003

De design compacto e funcional, o Alfa 3003 linha Plus, da Dismac, trabalha com até oito módulos de processamento escravo, que garantem desempenho superior ao de qualquer sistema multiusuário.

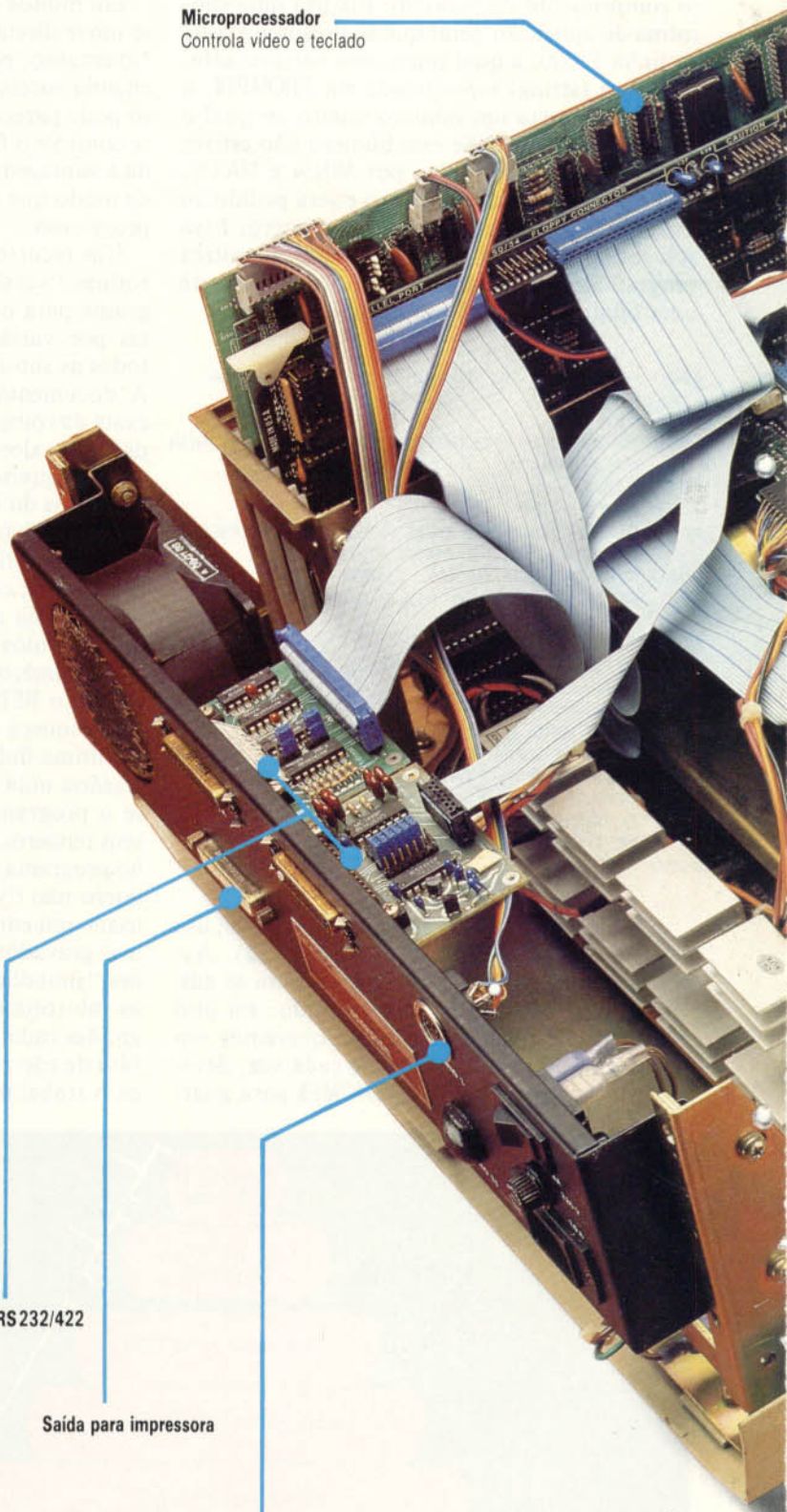
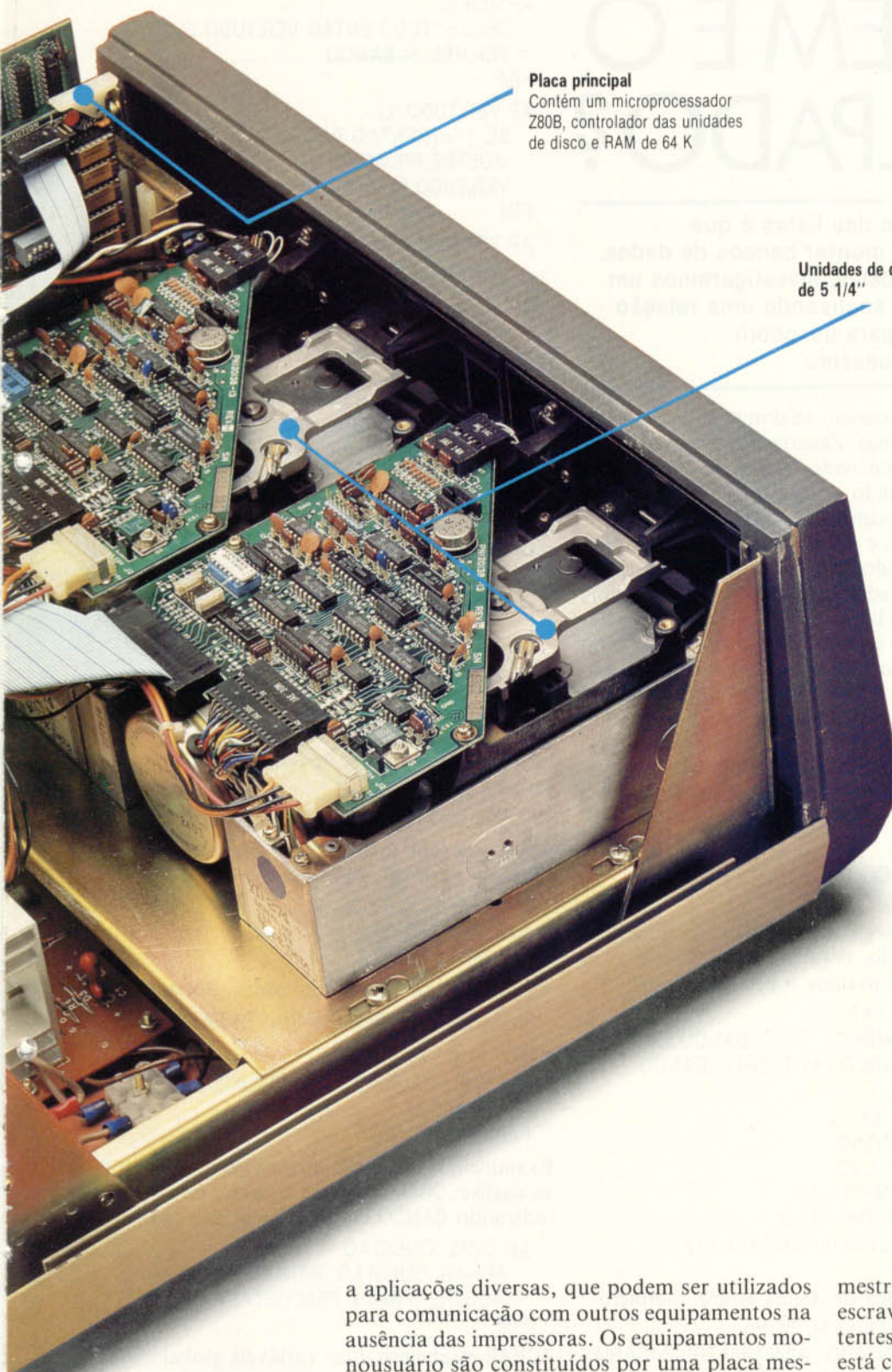
A linha Plus compreende uma série de equipamentos para aplicações profissionais, incorporando o que há de mais moderno em arquitetura de 8 bits. Esses equipamentos estão dispostos em configurações monousuário e multiusuário (vários terminais, cada um com sua própria unidade de processamento escrava). O sistema aceita até oito módulos de processamento escravo em comunicação com a unidade central de processamento (CPU mestra). Essa unidade, com acesso direto à memória, possui um relógio em tempo real, o que facilita as aplicações utilizadas pelos terminais. Com esses dois recursos, os módulos de processamento escravo oferecem desempenho superior a qualquer outro sistema multiusuário.

As configurações monousuário suportam até quatro discos flexíveis e quatro rígidos, além de uma impressora paralela e duas seriais. Estas últimas ocupam canais de comunicação destinados

Microprocessador
Controla vídeo e teclado

Saída serial RS232/422

Saída para impressora

Saída adicional

Placa principal
Contém um microprocessador Z80B, controlador das unidades de disco e RAM de 64 K

Unidades de disco flexível de 5 1/4"

DISMAC ALFA 3003 LINHA PLUS

MICROPROCESSADORES

Mestre: Zilog Z80B.
Dedicado: Intel 8085.

CLOCK

6 MHz.

VÍDEO

Monitor de 12", fósforo verde, 24+1 linhas de oitenta caracteres.

MEMÓRIA

RAM de 128 K.
ROM de 8 K.

TECLADO

Alfanumérico e numérico reduzido. Tecnologia Reed.

PERIFÉRICOS

Impressoras seriais e paralelas (máximo de seis); unidades de disco flexível, dupla face, dupla densidade (máximo de quatro); unidades de disco rígido (máximo de quatro); interface serial RS232 ou 422; modems Ralf-duplex ou full-duplex; comunicação síncrona ou assíncrona.

LINGUAGENS

COBOL MB, BASIC MB e FORTRAN 80.

SISTEMA OPERACIONAL

Sistema operacional multiusuário Dismac Rede versão 1.4, compatível com CP/M e MP/M.

DOCUMENTAÇÃO

Manual de linguagem, manual do sistema operacional e manual do usuário.

a aplicações diversas, que podem ser utilizados para comunicação com outros equipamentos na ausência das impressoras. Os equipamentos monousuário são constituídos por uma placa mestra e uma placa controladora de vídeo e teclado.

As configurações multiusuário admitem uma impressora paralela e cinco canais de comunicação serial, em geral usados para impressoras adicionais, mas que também servem para comunicação com outros equipamentos. Tal como as monousuário, essas configurações também suportam até quatro discos flexíveis e quatro rígidos de capacidades distintas. Possuem uma placa mestra, um controlador de disco rígido e placas escravas em número igual ao dos terminais existentes. No modelo com dois terminais, um deles está conectado à placa mestra, o que exige apenas uma placa escrava.

Os terminais da rede podem ser ligados por cabos co-axiais à unidade central, distante até 1 km. O sistema aceita as linguagens COBOL MB, BASIC MB e FORTRAN 80.

O equipamento permite trabalhar com teleprocessamento e dispõe de protocolos para todas as máquinas de médio e grande portes existentes no mercado.

QUEM É O CULPADO?

Uma vantagem das listas é que nos permitem montar bancos de dados. Com a ajuda delas, investigaremos um terrível crime, analisando uma relação de suspeitos para descobrir quem foi o assassino.

Um terrível assassinato abalou uma pequena aldeia nas montanhas: Zacarias foi atacado e morto a golpes de machado. Sabemos que Mateus e José têm machado, que Jaime e Heitor têm revólver e que a prima Alice tem uma faca. Nas mãos de Mateus e Jaime havia sangue quando foram interrogados pelo delegado da região.

O banco de dados em LOGO sobre esse crime consistirá numa lista de *fatos* — cada um deles compreendendo uma *relação* associada a um ou mais substantivos. No LOGO, representa-se um fato assim: [POSSUI MATEUS MACHADO], ou, em linguagem comum: "Mateus possui um machado". Para dizer que Jaime tinha sangue nas mãos, usamos [ENSANGUENTADO JAIME].

No início da investigação, o banco de dados está vazio:

```
AP PREPARAR
  FACA "BANCO []
FIM
```

Os fatos serão registrados durante a investigação. Por exemplo, para incluir ADICIONE [POSSUI ALICE FACA] usamos o procedimento:

```
AP ADICIONE :FATO
  SE NAO MEMBRO? :FATO :BANCO ENTAO
    FACA "BANCO PENT :FATO :BANCO
FIM
AP MEMBRO? :L1 :L2
  SE :L2=[] ENTAO
    SAIDA "FALSO
  SE :L1=PRIMEIRO :L2
    ENTAO SAIDA "VERD
  MEMBRO? :L1 SEMPRIMEIRO :L12
FIM
```

No final, o banco de dados estará completo:

```
[[ENSANGUENTADO MATEUS]
  [ENSANGUENTADO JAIME] [MATOU ZACARIAS
  MACHADO] [POSSUI MATEUS MACHADO]
  [POSSUI JOSE MACHADO] [POSSUI JAIME
  REVOLVER] [POSSUI HEITOR REVOLVER]
  [POSSUI ALICE FACA]]
```

Para imprimir os dados armazenados use VER. Acrescentando "TUDO, serão mostrados todos os dados. No caso de acrescentar o nome de uma relação, somente os fatos associados a ela serão impressos, como em VER "POSSUI.

```
AP VER :S
  SE :S="TUDO ENTAO VER.TUDO :BANCO
  VER.REL :S :BANCO
FIM
AP VER.TUDO :LI
  SE :LI=[] ENTAO PARE
  MOSTRE PRIMEIRO :LI
  VER.TUDO SEMPRIMEIRO :LI
FIM
AP VER.REL :S :LI
  SE :LI=[] ENTAO PARE
  SE :S=PRIMEIRO PRIMEIRO :LI ENTAO
    MOSTRE PRIMEIRO :LI
  VER.REL :S SEMPRIMEIRO :LI
FIM
```

Agora precisamos criar métodos de consulta ao banco de dados. O tipo mais simples verifica se um fato é considerado verdadeiro. Para isso, usamos o procedimento VERIFIQUE, que informa se um fato está no banco de dados. Por exemplo, VERIFIQUE [POSSUI ALICE FACA] tem como resultado SIM.

```
AP VERIFIQUE :FATO
  SE MEMBRO? :FATO :BANCO ENTAO
    MOSTRE "SIM SENAO MOSTRE "NAO
FIM
```

Seria muito útil, nesta investigação, formular perguntas do tipo "Quem possui um machado?". Para fazer isso empregamos as "variáveis", iniciadas por ?. Podemos então recolocar assim a pergunta:

```
QUAL [POSSUI ?ALGUEM MACHADO]
```

A resposta será uma lista de todos os valores possíveis da variável ?ALGUEM.

```
[?ALGUEM MATEUS]
[?ALGUEM JOSE]
FIM DA RESPOSTA
```

As variáveis podem ser múltiplas. Por exemplo:

```
QUAL [MATOU ?HOMEM ?INSTRUMENTO]
```

resultará na resposta:

```
[?HOMEM ZACARIAS] [?INSTRUMENTO
  MACHADO]
FIM DA RESPOSTA
```

Examinemos os procedimentos que permitem essa análise. QUAL transfere a tarefa para ACHAR, indicando BANCO como a fonte dos fatos.

```
AP QUAL :QUESTAO
  ACHAR :QUESTAO :BANCO
  MOSTRE [FIM DA RESPOSTA]
FIM
```

ACHAR estabelece duas variáveis globais, VARS e ANS. VARS serve para armazenar todos os possíveis conjuntos de valores das variáveis na pergunta, os quais são reunidos na lista ANS.

```
AP ACHAR :QUESTAO :DADOS
  FACA "VARS []
  FACA "ANS []
  COMPARAR :QUESTAO :DADOS
    MOSTREL :ANS
FIM
```

COMPARAR examina cada um dos fatos e, se houver alguma correspondência entre eles, os novos valores em VARS serão acrescentados a ANS. COMPARAR volta a associar VARS à lista vazia, e continua a procurar outras correspondências.

```
AP COMPARAR :QUESTAO :DADOS
  SE :DADOS = [] ENTAO PARE
  SE IGUAL? :QUESTAO PRIMEIRO :DADOS
    ENTAO FACA "ANS PENT :VARS :ANS
  FACA "VARS []
  COMPARAR :QUESTAO SEMPRIMEIRO :DADOS
FIM
```

Suponha que as entradas para IGUAL? sejam [POSSUI ?ALGUEM MACHADO] e [POSSUI JOSE MACHADO]. Nesse caso, IGUAL? tem VERDADEIRO como saída e atribui [?ALGUEM JOSE] a VARS. Se as entradas forem [POSSUI ?ALGUEM MACHADO] e [MATOU ZACARIAS MACHADO], então IGUAL? terá FALSO como saída.

Entretanto, a dificuldade surge quando há mais de uma variável envolvida. VALE? verifica se já foi atribuído algum valor para a variável associada a determinado fato.

Empregaremos uma notação alternativa para comandos condicionais. Se o resultado de TESTE for verdadeiro, as ações indicadas por SEVERD serão executadas; caso contrário, executam-se as ações indicadas por SEFALSO.

```
AP IGUAL? :QUESTAO :FATO
  SE [SETODOS [:QUESTAO = []][:FATO = []]]
    ENTAO SAIDA "VERD
  TESTE PRIMEIRO PRIMEIRO :QUESTAO = "?
  SEVERD SE NAO VALE? PRIMEIRO :QUESTAO
    PRIMEIRO :FATO :VARS ENTAO SAIDA
    "FALSO
  SEFALSO SE NAO [PRIMEIRO :QUESTAO =
    PRIMEIRO :FATO] ENTAO SAIDA "FALSO
  SAIDA IGUAL? SEMPRIMEIRO :QUESTAO
    SEMPRIMEIRO :FATO
FIM
```

Para entender o funcionamento de VALE?, veja o caso em que seus dados são ?INSTRUMENTO, MACHADO e ?HOMEM ZACARIAS. VALE? verifica se INSTRUMENTO pode ter o valor MACHADO. Há três possibilidades: ?INSTRUMENTO possui um valor, que não é MACHADO, fazendo com que a saída de VALE? seja FALSO; ou, então, ?INSTRUMENTO possui o valor MACHADO, e VALE? tem como resultado VERD; ou, ainda, ?INSTRUMENTO não tem um valor, recebendo o valor MACHADO. Essa informação vai para a VARS e a saída é VERD.

```
AP VALE? :NOME :VALORV :LISTAV
  SE :LISTAV = [] ENTAO FACA "VARS
    UENT LISTA :NOME :VALORV :VARS
    SAIDA "VERD
  TESTE :NOME = PRIMEIRO PRIMEIRO
    :LISTAV
  SEVERD SE :VALORV = ULTIMO PRIMEIRO
    :LISTAV ENTAO SAIDA "VERD SENAO
    SAIDA "FALSO
  SAIDA VALE? :NOME :VALOR SEMPRIMEIRO
    :LISTAV
FIM
```

MOSTREL simplesmente faz com que os itens de ANS sejam impressos um abaixo do outro.

```
AP MOSTREL :LISTAV
  SE :LISTAV = [] ENTAO PARE
  MOSTRE PRIMEIRO :LISTAV
  MOSTREL SEMPRIMEIRO :LISTAV
FIM
```

Perguntas mais complexas

No entanto, a investigação sobre o assassinato dificilmente avançará se não pudermos fazer perguntas mais complexas, do tipo "Que instrumento matou Zacarias e quem possui tal instrumento?". A tradução disso para o LOGO seria:

```
QUAL [MATOU ZACARIAS ?INSTRUMENTO]
[POSSUI ?SUSPEITO ?INSTRUMENTO]
```

Assim, os dados de entrada para QUAL são uma lista de perguntas e os valores encontrados serão aqueles que tornam verdadeiras todas as perguntas. Uma pergunta simples com esse novo tipo de QUAL utiliza a seguinte sintaxe:

```
QUAL [[POSSUI ?UMA FACA]]
```

Precisamos alterar um pouco os procedimentos:

```
AP QUAL :QUESTAO
  ACHAR :QUESTAO :BANCO
  MOSTRE [FIM DA RESPOSTA]
FIM
AP ACHAR :QUESTAO :DADOS
  FACA "VARS []
  FACA "ANS []
  COMPARAR :QUESTAO :DADOS
  MOSTREL :ANS
FIM
```

A função de COMPARAR tornou-se agora mais complicada. Tome [[MATOU ZACARIAS ?INSTRUMENTO][POSSUI ?SUSPEITO ?INSTRUMENTO]] como um exemplo de entrada. COMPARAR examina o banco para encontrar a resposta da primeira pergunta, e acaba fazendo a associação de ?INSTRUMENTO com MACHADO. A rotina passa então para a segunda pergunta ([POSSUI ?SUSPEITO ?INSTRUMENTO]) e repete o processo. Na segunda condição, ela encontra uma correspondência entre o valor de ?INSTRUMENTO — MACHADO e o de ?SUSPEITO — MATEUS. Como não há mais perguntas, essa é uma solução possível.

No entanto, pode haver outros valores que satisfaçam a segunda pergunta, mantendo MACHADO como o valor de ?INSTRUMENTO. Desse modo, COMPARAR retoma o exame dos dados a partir do ponto em que parou, e de fato encontra uma segunda solução com ?SUSPEITO como JOSE. O procedimento não pára aí, mas continua pesquisando em BANCO. Agora, ele chega ao fim sem encontrar outros valores que possam ser associados.

Uma vez que é possível outra solução para a primeira pergunta — que não seja MACHADO —, devemos retornar ao ponto em que foi encon-

Matriz de assassinato

O sr. Pereira foi encontrado morto, com trinta golpes de formão, na carroçaria de um caminhão. A polícia identificou quatro suspeitos: um pedreiro, um açougueiro, uma jardineira e um desenhista. Todos tinham acesso a instrumentos cortantes: a faca do açougueiro, a tesoura da jardineira, o estilete do desenhista e o formão do pedreiro. Um deles foi visto na esquina de uma rua, outro no galpão de um jardim e um terceiro alegou estar dormindo em casa. Aquele que puder ser vinculado ao caminhão é o assassino.
A investigação policial revelou os seguintes fatos:
1) Nenhum deles estava com sua ferramenta de trabalho na noite do crime. **2)** O açougueiro foi localizado no galpão abrindo cartas com um estilete. **3)** Uma testemunha ocular confirmou que o pedreiro se encontrava na esquina de uma rua, onde depois a jardineira acharia sua tesoura. **4)** A jardineira estava em sua cama usando uma faca de açougueiro para preparar um sanduíche de carne.

trada essa correspondência e prosseguir a partir daí. Esse processo é chamado retrocesso. Nesse caso, não há, de fato, outras soluções.

Para não perder o ponto em que se encontra o processo de atribuição de variáveis, COMPARAR coloca os valores atuais em uma pilha — antes de IGUAL? ser usada, já que ela pode alterar as atribuições — para recuperá-los depois. Eis o procedimento completo:

```
AP COMPARAR :QUESTAO :DADOS
  SE QUESTAO = [] ENTAO FACA "ANS
    PENT :VARS :ANS PARE
  SE DADOS = [] ENTAO PARE
  GUARDE :VARS
  TESTE IGUAL? PRIMEIRO :QUESTAO
    PRIMEIRO :DADOS
  SEVERD COMPARAR SEMPRIMEIRO
    :QUESTAO :DADOS
  TRAGA "VARS
  COMPARAR :QUESTAO SEMPRIMEIRO
    :DADOS
FIM
```

Em COMPARAR empregamos uma pilha para manter o controle sobre os valores de VARS, pois, se usássemos uma variável temporária, COMPARAR poderia chamar a si mesma entre o momento de gravar os valores e o de utilizá-los. Sem a pilha, qualquer variável temporária desse tipo poderia ser sobreposta pela chamada seguinte e os valores originais seriam perdidos.

GUARDE coloca um valor no "topo" da pilha, criando antes a variável PILHA, caso ela não exista.

```
AP GUARDE :DADOS
  SE NAO VALOR? :PILHA ENTAO
    FACA "PILHA []
  FACA "PILHA PENT :DADOS :PILHA
FIM
```

TRAGA tira um item da pilha e o atribui como valor de uma variável.

```
AP TRAGA :NOME
  FACA :NOME PRIMEIRO :PILHA
  FACA "PILHA SEMPRIMEIRO :PILHA
FIM
```

São esses os rudimentos de uma linguagem de "programação lógica", em que simplesmente acrescentamos os fatos e as regras a um banco de dados e então o consultamos por meio de descrições lógicas dos dados de que precisamos.

TRILHAS ESPIRAIS

À medida que cresce a produção de discos flexíveis e videodiscos, esses dispositivos se tornam mais acessíveis a usuários domésticos e a pequenas empresas, com toda a sua variedade de aplicações.

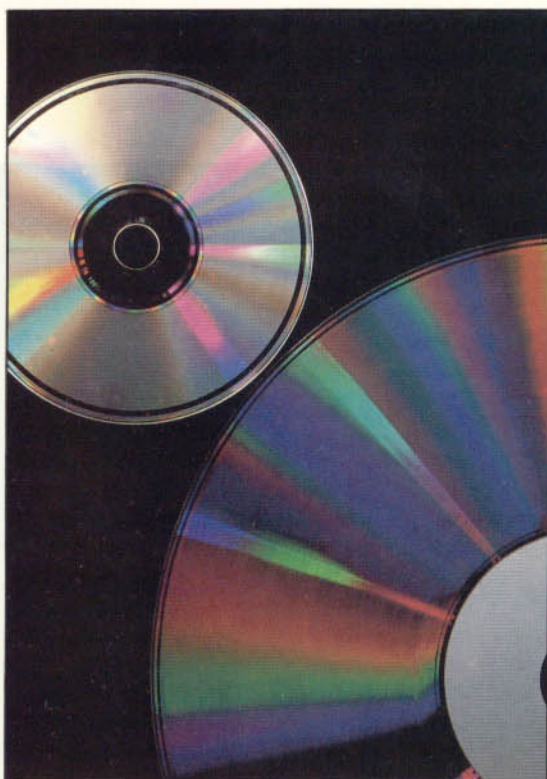
Em muitos países, os gravadores de vídeo com discos a laser, considerados artigos de luxo até há poucos anos, já custam menos que os gravadores de videocassete, mesmo proporcionando melhor reprodução de imagem.

Uma cena normal de televisão é uma imagem dinâmica que pode ser gravada em videoteipe como seqüência ininterrupta. A única maneira de localizar determinada parte da seqüência gravada é movendo a fita para a frente, em velocidade normal ou acelerada, ou usando a tecla de retrocesso.

Os discos armazenam as imagens em quadros separados, sendo possível ir diretamente e com precisão a qualquer um deles. A localização de uma imagem no disco pode ser descrita em termos de trilha e setor, e um microprocessador mantém um catálogo atualizado do endereço de cada uma. O microprocessador supervisiona o acesso ao disco e permite congelar a imagem ou passá-la em câmara lenta e som estereofônico.

Tarefas tais como girar um prato giratório em velocidade alta e constante e posicionar uma cabeça de leitura exatamente em determinado lugar da superfície não são as realizações mais surpreendentes da tecnologia do disco. O maior desafio consistiu em obter a altíssima densidade de armazenamento dos discos, com o tamanho de um LP de 12 polegadas.

Se você já usou gráficos de alta resolução em seu micro, sabe que a imagem da televisão é composta de pontos (pixels) de luz isolados — quanto mais pontos houver na tela, melhor a imagem apresentada — e que o armazenamento de telas de alta resolução necessita de grande quantidade de memória.

O laser chega ao lar
Os toca-discos a laser tendem a ser tão baratos quanto os microcomputadores, e não será raro encontrar os dois interligados em casa. Adquirindo os discos e o software apropriados, você poderá ter acesso a bancos de dados pictóricos e a sofisticados programas de treinamento, bem como a jogos de aventura.

Memória inovadora
Discos a laser de 12 polegadas e compactos menores servem para armazenar informações de vídeo e áudio, além dos dados digitais hoje gravados em discos flexíveis e fitas cassete. A desvantagem é que não se pode gravar nesses discos e tecnicamente eles são apenas ROM. Assim, você não grava seus próprios programas e informações em discos a laser; deve utilizar software pré-gravado.

Se uma tela de televisão tiver resolução de 640 x 256 pixels (equivalente à dos melhores micros), então o armazenamento de um segundo de vídeo (a 25 quadros por segundo) exigirá 25 x 20 Kbytes de memória.

Um minuto de um programa de televisão ocuparia 30 Mbytes (30 megabytes, ou seja, 30 milhões de bytes) e um episódio de um seriado preferido chegaria a ocupar 1 Gbyte (1 gigabyte, ou seja, 1.000 megabytes) de memória.

Se você utilizasse discos flexíveis de densidade simples e face simples, usados na maioria dos microcomputadores, seriam necessários mais de 6.000 discos e uma semana de trabalho!

A resposta a esse desafio consiste em gravar os dados de maneira diminuta. Um gravador de disco a laser grava os dados com um raio laser da espessura de um fio de cabelo sobre uma placa metálica revestida com uma camada rígida translúcida. Um raio de baixa potência é usado para ler os dados no disco.

Os lasers são usados como uma agulha de leitura-gravação para esse tipo de disco porque são dispositivos de baixa tolerância e fina resolução. Nenhuma outra técnica conhecida pode ler e gravar com eficiência tantos dados em tão pouco espaço.

O formato é uma combinação das técnicas usadas em discos musicais e disquetes flexíveis. Os sulcos de um disco de música formam uma espiral contínua e as paredes dos sulcos contêm uma representação gravada das formas de onda do som.

A maioria dos usuários de micro sabe que as trilhas do disco flexível são círculos concêntricos e que a informação é gravada magneticamente sobre o disco e armazenada de modo digital em padrões de 0 e 1.

O formato do disco laser usa trilhas e não sulcos, mas em forma de espiral. A informação é gravada opticamente sobre o disco em padrões de 0 e 1, mas não pode ser apagada. Os dígitos 0 e 1 (que representam os padrões de pontos que compõem uma imagem de televisão) são gravados sobre a superfície do disco pelos minúsculos orifícios que o laser faz queimando a película de metal para representar os dígitos 1 e deixando-a intata para representar os 0. Cada orifício tem 0,5 μm (meio micrômetro, ou seja, 0,0005 mm) de largura e 0,1 μm (um décimo de micrômetro, ou seja, 0,0001 mm) de profundidade. Portanto, 1 cm² do disco pode conter 400 milhões desses orifícios.

Essa miniaturização extraordinária é suficiente para suprir a demanda de armazenamento de vídeo. Um disco de 35 cm de diâmetro contém 54.000 quadros de televisão por lado, ou aproximadamente 36 minutos de gravação. Os cálculos anteriores referiam-se à resolução de gráficos de computador em preto e branco, enquanto o disco deve armazenar uma trilha de áudio e também informações sobre as cores de cada ponto da imagem da televisão.

Um quadro colorido, com sua respectiva trilha sonora, pode precisar de 100 Kbytes de espaço para armazenamento. Para 54.000 quadros desse tipo, seria necessário usar 5,4 milhões de Kbytes, ou 5,4 Gbytes.

Tendo resolvido o problema de limitação de armazenamento, os discos laser oferecem grandes possibilidades de aplicação. Um benefício importante consiste em remover um dos maiores obstáculos em processamento de dados — coleta e entrada de informação. Mesmo assim, é necessário sentar diante de um terminal e digitar representações codificadas da informação para o sistema.

Se, ao invés disso, você puder apontar uma câmara para os dados e deixar que ela armazene a informação visual no disco enquanto você digita apenas os detalhes dos índices das imagens gravadas, a carga de trabalho apresentará notável diminuição.

Os toca-discos a laser variam muito no que se refere à sofisticação. Já se encontram disponíveis gravadores de vídeo com discos a laser para uso doméstico. Eles passam imagens como um gravador comum de videocassete, mas com a vantagem adicional de oferecer alta definição tanto em velocidade normal quanto em câmara lenta. Você também pode escolher quadros isolados em alguns equipamentos simplesmente digitando seu número num controle manual. Os que fazem isso — fabricados por empresas co-

mo a Pioneer e a Philips — ainda são caros e de uso exclusivamente profissional.

O sistema mais simples consiste em usar uma interface IEEE ou RS232 para que o computador selecione um determinado quadro pelo seu número. Um software apropriado mantém uma lista dos quadros disponíveis num banco de dados e os seleciona quando necessário.

A Philips levou a idéia um passo adiante e incorporou um microcomputador simples a seus modelos mais recentes. Assim, pode-se carregar um programa para determinado disco a partir de um cartucho EPROM conectado à máquina ou então a partir do próprio disco a laser. Cada disco a laser armazena duas trilhas de áudio e uma de vídeo. Isso permite que um único disco tenha trilha sonora em duas línguas, por exemplo. Contudo, se a segunda trilha de áudio não for necessária, poderá ser usada para armazenar um programa de computador.

Assim, temos um banco de 54.000 imagens de qualidade sob controle do computador. O estágio final seria combinar as imagens do disco a laser e o texto do computador. Isso poderia ser feito com dois monitores separados, ou combinando as duas entradas de vídeo, ou ainda usando-se um monitor com seu próprio gerador de teletexto.

Surge assim um meio de comunicação inteiramente novo — o vídeo interativo. O usuário e o software podem dirigir o que aparece na tela, tanto as cenas de ação quanto as imagens congeladas, lendo o disco em qualquer ordem.

Sua aplicação mais óbvia é para um banco de dados pictórico. O usuário pode fazer perguntas ao computador, que retira as informações relevantes de um banco de dados e instrui o equipamento a apresentar o quadro de vídeo apropriado. As vantagens para efetuar consultas em bibliotecas e escolas são evidentes.

O vídeo interativo já avançou um estágio, envolvendo o usuário nas seqüências mostradas na tela: um programa de treinamento explica alguma coisa com um pequeno trecho de um filme e então faz perguntas, recapitulando se houver necessidade, ou aprofundando-se nos detalhes.

O usuário pode até modificar o roteiro e o final de um filme pronto, "dirigindo-o", ou ampliar as possibilidades dos jogos de aventura.

Além do problema do custo do equipamento e da fabricação dos discos a laser, novos recursos precisam ser desenvolvidos no planejamento e na produção de discos interativos, tanto em termos de software de computador quanto na execução do roteiro e da filmagem das cenas.

Muitas empresas pequenas em todo o mundo estão começando a enfrentar esses desafios e uma multinacional, a Computer Assisted Televideo (CAT), já oferecia, em 1985, um serviço completo para clientes — escolha e instalação do equipamento, planejamento e produção de discos e criação de software adequado. Suas atividades, porém, se limitaram à produção de programas de treinamento.

No entanto, com o tempo, os discos a laser poderiam ser vendidos, com software, por preço um pouco superior ao dos filmes e videoteipes. Considerando a qualidade dos programas de entretenimento e educativos por eles possibilitados, o mercado, sem dúvida, os absorveria.

Vídeo interativo
Muitos toca-discos a laser possuem uma interface IEEE ou RS232, que lhes permite ser controlados por microcomputadores.
Um conjunto típico seria um computador com um banco de dados armazenado em discos flexíveis e relacionado com as imagens no disco a laser.
O usuário escolhe um item no banco de dados e, a seguir, o computador faz com que o toca-discos a laser selecione e exiba a imagem apropriada, tendo como referência o número do quadro. O sistema aqui mostrado adota o recurso de combinar a saída do computador com as imagens do disco a laser na mesma tela.

SIGA O MAPA

Muito úteis na simplificação dos circuitos lógicos, os mapas de Karnaugh não substituem todas as simplificações algébricas, mas reduzem o esforço para fatorar expressões complexas.

Os mapas de Karnaugh (também conhecidos como "mapas-k") são na verdade extensões dos diagramas de Venn, já examinados anteriormente.

Esses diagramas possibilitam representar expressões lógicas por meio de figuras. Já um mapa-k assume formas diversas, dependendo do número de letras (ou variáveis) diferentes que houver na expressão a se simplificar.

O mapa-k é mais útil para expressões que contêm duas, três ou quatro variáveis.

Duas variáveis. Cada quadrado de um mapa-k de duas variáveis (há $2^2 = 4$ quadrados) representa uma função E, como se mostra abaixo:

	A	$\overline{A}$
B	A.B	$\overline{A}$.B
$\overline{B}$	A.$\overline{B}$	$\overline{A}$.$\overline{B}$

Representa-se a expressão $AB + A\overline{B}$ como um mapa-k, colocando os 1 nos quadrados próprios:

	A	$\overline{A}$
B	1	0
$\overline{B}$	1	0

Aqui estão três outros exemplos que representam as expressões $\overline{A}\overline{B}$, $AB + \overline{A}\overline{B}$ e $AB + \overline{A}B + A\overline{B}$, respectivamente:

Três variáveis. Nesse caso, o número de quadrados aumenta em um fator de dois ($2^3 = 8$ quadrados). O mapa-k básico é:

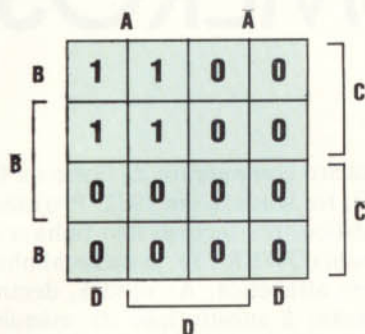
Eis duas expressões, $AC + \overline{A}B\overline{C}$ e $AB + \overline{A}C$, representadas como mapas-k:

$$ABC + A\overline{B}C + \overline{A}B\overline{C} =$$
$$= AC(B + \overline{B}) + \overline{A}B\overline{C}$$
$$= AC + \overline{A}B\overline{C}$$

$$ABC + AB\overline{C} + \overline{A}BC + \overline{A}\overline{B}C =$$
$$= AB(C + \overline{C}) + \overline{A}C(B + \overline{B})$$
$$= AB + \overline{A}C$$

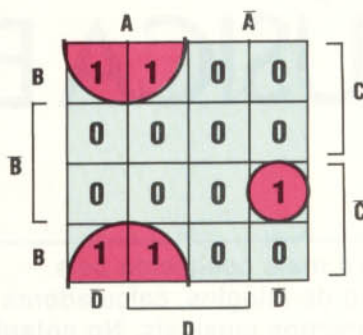
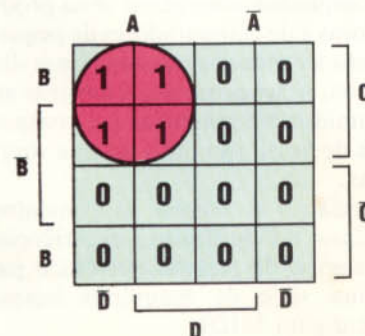
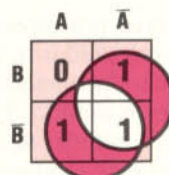
Ambas as expressões foram simplificadas usando-se a lei booleana que estabelece que um conjunto A operado em OU com sua negativa ($\overline{A}$) produz 1 — o conjunto Universo ou Identidade.

Quatro variáveis. Os mapas se tornam mais complexos (eles têm $2^4 = 16$ quadrados), mas mesmo assim são bastante simples de interpretar, de acordo com o diagrama básico:

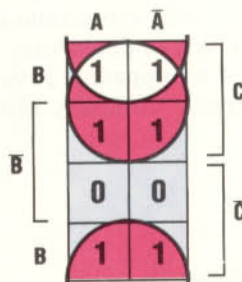
Observe agora um mapa-k com simplificação correspondente:

$$\begin{aligned}&AB C\overline{D} + ABCD + A\overline{B}C\overline{D} + A\overline{B}CD =\\ &= ABC(\overline{D} + D) + A\overline{B}C(\overline{D} + D)\\ &= ABC + A\overline{B}C\\ &= AC(B + \overline{B})\\ &= AC\end{aligned}$$

Eis aqui outro exemplo:

$$\begin{aligned}&ABC\overline{D} + ABCD + \overline{ABCD} + AB\overline{CD} + AB\overline{C}D\\ &= ABC(\overline{D} + D) + \overline{A}\,\overline{BCD} + AB\overline{C}(\overline{D} + D)\\ &= ABC + \overline{ABCD} + AB\overline{C}\\ &= AB(C + \overline{C}) + \overline{ABCD}\\ &= AB + \overline{ABCD}\end{aligned}$$

O exame atento da disposição dos 1 nos dois mapas-k anteriores revela certos padrões. No primeiro exemplo, todos os quadrados com AC em suas expressões estão preenchidos com 1. No segundo exemplo, isso acontece com todos os quadrados com AB. Daí, um método mais fácil de simplificar expressões booleanas consiste em inspecionar um mapa-k. Considere:

Com um pouco de prática, podem-se escolher grupos de dois, quatro ou oito 1 para formar termos mais simples. Por exemplo, examinemos a expressão: $\overline{A}B + A\overline{B} + \overline{AB}$.

Usando um mapa-k de duas variáveis, podemos escolher dois grupos de 1. O primeiro representa todas as ocorrências de NÃO(B) e o outro todas as ocorrências de NÃO(A), de modo que a expressão é simplificada para $\overline{A} + \overline{B}$, ou, lembrando a lei de Morgan, para: $\overline{A.B}$. É possível chegar a essa conclusão mais diretamente examinando o mapa-k?

Um exemplo mais difícil envolve uma expressão de três variáveis:

$$ABC + A\overline{B}C + \overline{A}BC + AB\overline{C} + \overline{A}B\overline{C} + \overline{AB}C$$

O grupo de quatro 1 na parte de cima do mapa-k representa todos os casos possíveis em que C é verdadeira. A primeira e última carreiras do mapa representam todos os casos possíveis em que B é verdadeira. Assim, a expressão simplificada será: B + C.

No próximo artigo da série trataremos do uso dos mapas de Karnaugh na simplificação de expressões booleanas com quatro variáveis. Então, ao mostrar como os mapas-k são usados no processo de desenhar circuitos, reuniremos todos os aspectos até agora apresentados.

EXERCÍCIO 4

Desenhe mapas-k com três variáveis para simplificar as seguintes expressões booleanas:

a) $\overline{A}.B.C + \overline{A}.\overline{B}.C + \overline{B}.\overline{C} + \overline{A}.B.\overline{C}$

b) $A.\overline{B}.\overline{C} + \overline{A}.B.\overline{C} + A.B.\overline{C}$

Resposta na próxima matéria desta seção.

MÚSICA E NÚMEROS

A Casio é mais conhecida pela produção de relógios, calculadoras e instrumentos musicais. No entanto, tem lançado vários computadores portáteis e de bolso para penetrar no mercado da informática.

Apesar de responsável por cerca de 50% do mercado mundial de calculadoras, no início da década de 80 a Casio tinha apenas 3.300 funcionários, espalhados pelo mundo. Em 1983, seu faturamento foi de 29 milhões de dólares, quantia irrisória para um grande produtor de equipamentos eletrônicos. Tony Manton, diretor de vendas de calculadoras da empresa, declarou que essa é uma atitude deliberada: "Somos uma companhia pequena e conservadora. Grandes campanhas publicitárias não fazem parte de nosso estilo".

A empresa foi fundada no Japão por cinço irmãos da família Kashio, após a Segunda Guerra Mundial (1939-45). Naquela época, era conhecida como Kashio Seisakujo e fabricava equipamentos de escritório. No início da década de 50, produziu a calculadora 14-A Relay, que pesava 130 kg e tinha o tamanho de uma mesa de escritório. Outras calculadoras foram fabricadas e em 1957 a companhia passou a chamar-se Casio Computer Company Ltd.

O Centro de Pesquisa e Desenvolvimento, em Tóquio.

Liderança mundial
Tadao Kashio, presidente da Casio, é um dos cinco membros da família Kashio que estão no topo da estrutura da empresa.

O primeiro computador de bolso da Casio, o FX-702P, foi lançado em 1982. Projetado para uso científico, seu teclado não tinha a disposição comum (QWERTY): as teclas alinhavam-se em ordem alfabética. As vendas, desanimadoras, levaram à substituição da máquina pela FX-700P, com teclado QWERTY.

Seguiu-se o PB-100, computador de bolso para uso comercial. Seu design assemelhava-se ao do FX-702P. Possuía, ainda, um visor de cristal líquido (LCD) e teclas numéricas.

A Casio produziu, além disso, gravadores cassete, impressoras/plotters e cartões de RAM especiais para as duas máquinas.

Um substituto recente do PB-100 é o FX-750P. Esse computador tem dois slots para cartões de RAM. Neles são inseridas pequenas placas de metal (que têm o tamanho de uma caixa de fósforos) com capacidade de armazenamento de 4 Kbytes. Cada cartão possui uma pilha de 3 V que mantém o conteúdo da memória, mesmo quando é ele retirado do computador. Desse modo, os programas não se perdem e podem ser carregados para a máquina sempre que requeridos. As pilhas duram um ano. Quando há necessidade de trocá-las, os programas são recarregados a partir de fitas.

A Casio também produz o FP-200, computador portátil equipado com 8 Kbytes de RAM (capacidade que se amplia para 32 Kbytes). O FP-200 tem um LCD com 8 linhas de vinte caracteres e capacidade de resolução gráfica de 160 x 64 pixels.

Posteriormente, a Casio lançou o SL-800. Trata-se de uma calculadora de baixo custo, com peso e tamanho aproximados de um cartão de crédito. Essa forma delgada resulta de um processo de fabricação conhecido como "filming", no qual os componentes são impressos em filme laminado, em vez de serem soldados em placas de circuitos convencionais.

Em muitos países europeus e do Extremo Oriente, a Casio distribui computadores comerciais e microcomputadores padrão MSX. Em outros, a companhia concentrou-se na produção de calculadoras e de computadores de pequeno porte, visando à expansão regrada. Uma das metas da empresa, nesse processo, é mostrar ao público consumidor em potencial que computadores portáteis de bolso são mais do que simples calculadoras.

Com relação a futuros lançamentos nessa área, a Casio tem utilizado a experiência adquirida no campo do teclado eletrônico para produzir uma série de máquinas baseadas na interface digital MIDI.

PROCESSADOR DE IDÉIAS

```
Col>|A          |B          |C          |D          |E          |F           |
Lin+-------------------------------------------------------------------------
  1|   DATA        XEROX     CORREIO    CARTORIO  TRANSPORTE TELEX/FONE
  2|
  3|  31/07/84    19888.00     666.00    7575.00       66.00        66.00
  4|  02/08/84       44.00     444.00    5555.00   142424.00      2424.00
  5|                444.00
  6|                888.00
  7|               4444.00
  8|               4444.00    6676.00  787878.00   787878.00     78787.00
  9|                                                              6666.00
 10|                           444.00
 11|                     >     444.00<
 12|
 13|
 14|
 15|
  +-------------------------------------------------------------------------
[   DESPOK]  cursor:   C11       posicao:   C11     L-R

posicao||        tipo: numeric
dado   ||    conteudo: 444
                 edit: █
```

Oferecendo múltiplas possibilidades de uso, a planilha eletrônica funciona como gerador de idéias, e constitui valioso auxiliar no exame e na amostragem de informações.

Como os processadores de texto e os bancos de dados, as planilhas eletrônicas possuem recursos que raramente chegam a ser explorados pelos usuários. A maior parte das pessoas que trabalham com processadores de texto não utiliza seus comandos mais sofisticados; já os bancos de dados tendem a ser usados apenas como índices e sistemas de gerenciamento de arquivos — quase nunca para processamento de dados.

Poucos proprietários de microcomputadores vêem utilidade no uso de planilhas eletrônicas. Quase todos acreditam que programas desse tipo seriam desinteressantes e de utilidade prática restrita. Em geral, ocorre a associação da planilha aos usos financeiros e comerciais — aplicações que quase sempre intimidam os usuários domésticos de microcomputadores.

Subestima-se, assim, a importância da administração financeira do lar, e deixa-se de considerar que a planilha é um processador de idéias cujo emprego ficou comprometido com a imagem de "instrumento de contabilidade". Na verdade, a planilha eletrônica está para as idéias assim como o processador de textos está para a criatividade de quem escreve.

A planilha eletrônica é ao mesmo tempo um processador de texto e uma calculadora. O nome que recebeu deve-se menos à função que desempenha que a seu aspecto, pois divide-se em linhas e colunas, como uma planilha de contabilidade.

Funções combinadas
Uma das grandes vantagens no uso da planilha eletrônica está na possibilidade de realizar projeções. Combinando as funções de calculadora e processador de texto, esse programa permite introduzir os dados nas células com fórmulas e obter os resultados de todas as células recalculados com os novos dados.

Cálculo de notas

Formato
O comando FORMATO foi usado para estabelecer a largura da coluna D, alinhar as células de texto pela esquerda e mostrar os números com duas casas decimais.

Copiar
Com este comando, um bloco de células pode ser copiado em qualquer parte da planilha.

Peso
Multiplica a nota real para produzir uma nota ponderada.

	! C !D!	E !!	F !!	G !!	H !	!	J !!	K !!	L !!	M
1			AJUSTAMENTO	DE NOTAS						
2	********	********	********	********	********	*	********	********	********	********
3			NOTAS REAIS			*		NOTAS PONDERADAS		
4	********	********	********	********	********	*	********	********	********	********
5						*	.75	.86	.73	
6		Mat.	Port.	Hist.	MEDIA	*	Mat.	Port.	Hist.	MEDIA
7	Artur :	87.00	55.00	76.00	72.67	*	65.25	47.30	55.48	56.01
8	Bruno :	75.00	37.00	46.00	52.67	*	56.25	31.82	33.58	40.55
9	Carlos :	39.00	95.00	48.00	60.67	*	29.25	81.70	35.04	48.66
10	Danilo :	88.00	63.00	95.00	82.00	*	66.00	54.18	69.35	63.18
11	Edgard :	24.00	26.00	63.00	37.67	*	18.00	22.36	45.99	28.78
12	Fabio :	94.00	88.00	88.00	90.00	*	70.50	75.68	64.24	70.14
13	Gustavo:	61.00	46.00	65.00	57.33	*	45.75	39.56	47.45	44.25
14	========	========	========	========	========	=	========	========	========	========
15	MEDIA :	66.86	58.57	68.71	64.71	*	50.14	50.37	50.16	50.23
16						*				
17	********	********	********	********	********	*	********	********	********	********

Texto repetido
Digitando um asterisco nesta célula, o recurso de repetição de texto da planilha preencherá toda a linha com asteriscos.

Cálculo automático
Entra-se com uma fórmula apenas uma vez e copia-se com um só comando a fórmula para outras células.

Média
Calculada por um só comando: MEDIA (célula 1... célula n).

Para comparar o desempenho de seus alunos nas diversas matérias, o professor quer ponderar todos os resultados de exames de forma que a média em cada disciplina seja a mesma. Ele tem de experimentar vários pesos para cada uma, calculando e recalculando as notas — um trabalho tedioso e fadado a erros, que uma planilha eletrônica poderia fazer em minutos. Na planilha, tudo, menos as notas reais, é calculado automaticamente. Se o peso for mudado, resultará, em poucos segundos, nova coluna de resultados ponderados para cada matéria.

Os dados são dispostos em células que, assim como as de uma planilha feita no papel, têm vários empregos. Nessas células, entra-se com textos que serão mostrados na tela ou no papel; com dados numéricos para exibição e cálculo; ou com fórmulas matemáticas que operam com o conteúdo de outras células.

Uma vez que as fórmulas estejam em seu lugar, a planilha transforma-se num programa gerado pelo usuário, pronto para a introdução de dados. Sempre que se introduzem novos dados numéricos, algébricos ou textuais, todas as células com fórmulas são recalculadas, em ordem. Dessa forma, a planilha mantém-se atualizada.

Tais características permitem utilizar a planilha eletrônica para tarefas simples de diagramação de tela e de impressão. No caso, ela facilita a formatação e a impressão não só de cálculos simples — cuja realização dispensaria a ajuda da máquina, se não fossem tão maçantes —, como também de operações complexas, nem imaginadas pelo usuário.

Em muitos casos, o uso da planilha revela algumas aplicações em que o proprietário do micro nunca havia pensado. Incluem-se aí atualização de estoques, análise de resultados esportivos e lotéricos, desenho de formas, produção de mapas de sincronização para som e luz em teatro, cálculo de impostos e até questões que exigem decisão, como optar entre a compra ou o aluguel de um televisor, considerando preço, utilização, desgaste e outros fatores.

Todas essas tarefas poderiam ser programadas por pessoas com conhecimentos de BASIC, mas cada uma delas levaria horas para ser desenvolvida. E a maior parte desse tempo seria despendida na solução e depuração dos infindáveis comandos PRINT TAB, PRINT AT e INPUT, essenciais na formatação das telas. A grande vantagem das planilhas é a possibilidade de formatar a tela à medida que se trabalha na relação entre as variáveis; ou seja, a tela é formatada de maneira simples, como se você estivesse escrevendo as informações numa folha de papel, dispondo textos, dados e resultados de cálculos nos lugares desejados.

Versões e opções

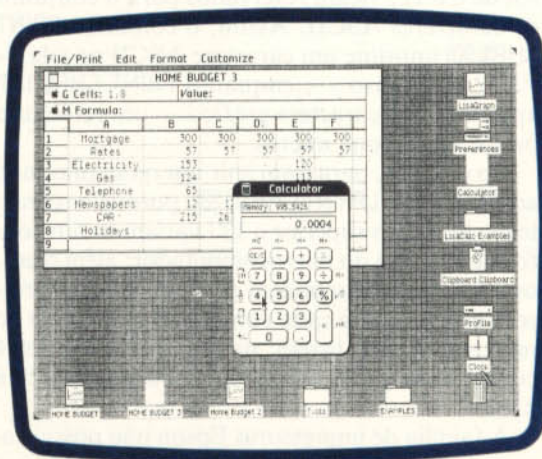
As planilhas eletrônicas possuem vários comandos que facilitam a formatação de tela. Você pode mover, copiar ou eliminar blocos de células, inserir ou eliminar colunas e linhas e definir o formato de uma célula ou bloco quanto a tamanho, justificação (alinhamento em relação aos outros itens da coluna) e posição da vírgula decimal. Na maioria das versões do BASIC, é difícil lidar com esses detalhes; no entanto, eles constituem operações essenciais na produção de relatórios, com as vantagens da facilidade de uso e da boa apresentação visual que oferecem.

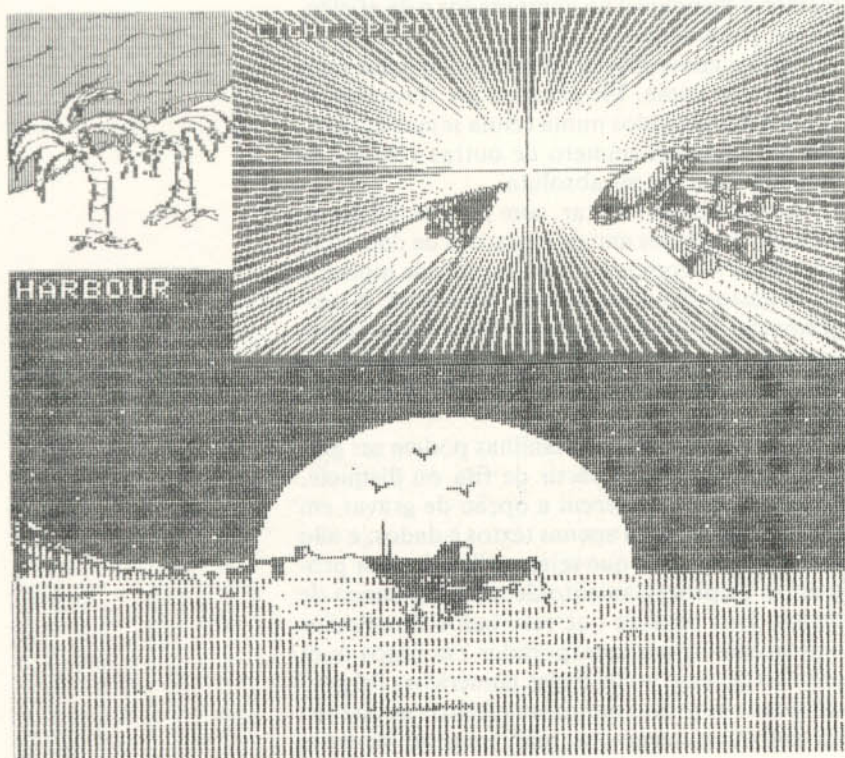
As funções de cálculo são análogas aos recursos para formatação. Com um único comando, calcula-se o valor médio de uma linha ou coluna de dados, contam-se as entradas não zero numa tabela. Somam-se os valores de uma matriz, encontram-se os valores máximo e mínimo numa lista e utilizam-se esses recursos em expressões matemáticas com funções e operadores mais conhecidos, como +, /, SQR (raiz quadrada) e ABS (valor absoluto). Mas nem todas as planilhas eletrônicas oferecem tantas e tão variadas possibilidades. As opções dependem muito da memória disponível no computador e da eficiência do programa.

Um dos comandos de planilha eletrônica mais úteis é o de cópia. Ele permite que fórmulas ou valores introduzidos numa célula sejam duplicados em qualquer número de outras células, de maneira relativa ou absoluta.

Isso possibilita montar, com poucos toques no teclado, projeções anuais e quadros de dados cumulativos — como os juros mensais de hipoteca ou os gastos semanais de uma residência. A programação de uma planilha torna viável a representação de expressões matemáticas complicadas de forma muito mais direta do que o BASIC permitiria.

Uma vez prontas, as planilhas podem ser gravadas e acessadas a partir de fita ou disquete. Muitas versões oferecem a opção de gravar em formato de arquivo apenas textos e dados, e não fórmulas, a fim de que sejam utilizados por programas de processamento de texto ou banco de dados. Isso permite que resultados de cálculos e projeções sejam incorporados em bloco num arquivo de texto ou de dados numéricos. Tal possibilidade constitui um passo efetivo em direção aos softwares integrados, mas, em geral, só existe nos programas mais caros.

As planilhas que dispõem de um conjunto razoável de comandos mostram-se tão grandes quanto a imaginação do usuário e a memória disponível no computador. Os próprios programas são, quase sempre, extensos; além disso, as aplicações com tabelas muito longas e os recursos sofisticados de processamento dos dados podem preencher rapidamente o restante da memória disponível. Deve-se levar em conta, ainda, que cálculos mais complicados reduzem a velocidade do programa.

Transferindo dados
No LisaCalc, planilha da Apple para o Lisa, é possível chamar à tela, por meio do mouse, várias "janelas", facilitando assim a transferência de dados de uma célula para outra.

PRIMEIRAS IMPRESSÕES

Impressão de tela
Estes desenhos foram feitos na tela usando-se um tablete gráfico. Depois, o conteúdo da tela foi reproduzido por uma impressora Epson FX-80, mostrando as possibilidades gráficas da impressora do tipo matricial.

A maioria dos usuários desconhece as possibilidades das impressoras matriciais. No entanto, elas permitem a produção de gráficos atraentes, por meio de um programa de dump de tela.

A maioria dos microcomputadores tem um modo gráfico de baixa resolução; isso porque os caracteres gráficos são do mesmo tamanho que os de texto. Os códigos desses caracteres de "bloco" são superiores a 127, uma vez que os números de 0 a 127 ficam reservados para o conjunto de caracteres ASCII. Assim, o comando PRINT CHR$(90) imprime um caractere ASCII na tela — "Z", nesse caso —, enquanto PRINT CHR$(128) apresenta um caractere gráfico — um retângulo preto, em alguns micros.

No entanto, para imprimir um retângulo, não adianta teclar LPRINT CHR$(128), porque os caracteres de código superior a 127 variam muito conforme a marca do microcomputador; além disso, os fabricantes não podem produzir uma impressora especial para cada computador existente no mercado. O que fazem, na maioria das vezes, é copiar o conjunto ASCII nos códigos de 128 a 255.

A família de impressoras Epson não possui caracteres gráficos; no entanto, você pode alterar qualquer dos caracteres ASCII a fim de produzir seus próprios caracteres gráficos. Para tanto, basta enviar códigos adequados à impressora.

Os gráficos de computadores de alta resolução são formados na tela por pequenos pontos (ou pixels) e não por caracteres inteiros. Da mesma forma, a impressão de alta resolução em papel usa pequenos pontos de tinta. A cabeça de impressão numa impressora matricial tem diversas agulhas arranjadas numa linha vertical que se move de um lado para outro do papel enquanto imprime. Em geral, os caracteres são formados por uma matriz de agulhas (muitas vezes de oito por oito pontos). No entanto, é possível produzir gráficos controlando as agulhas uma a uma.

O primeiro passo consiste em ligar a impressora no modo gráfico. Como acontece com qualquer outro tipo de impressão, isso é feito enviando um código específico para o modelo de impressora. Na Epson FX-80, por exemplo, as instruções necessárias são:

```
LPRINT CHR$(27);"K";CHR$(N1);(N2);
```

A letra "K" indica o modo gráfico, e os números (N1) e (N2) determinam a largura de cada linha de gráfico — em outras palavras, o número de pontos, feitos pelas agulhas, que cabem de um lado a outro da página.

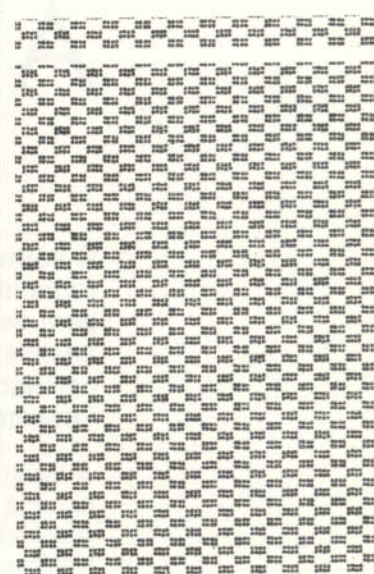
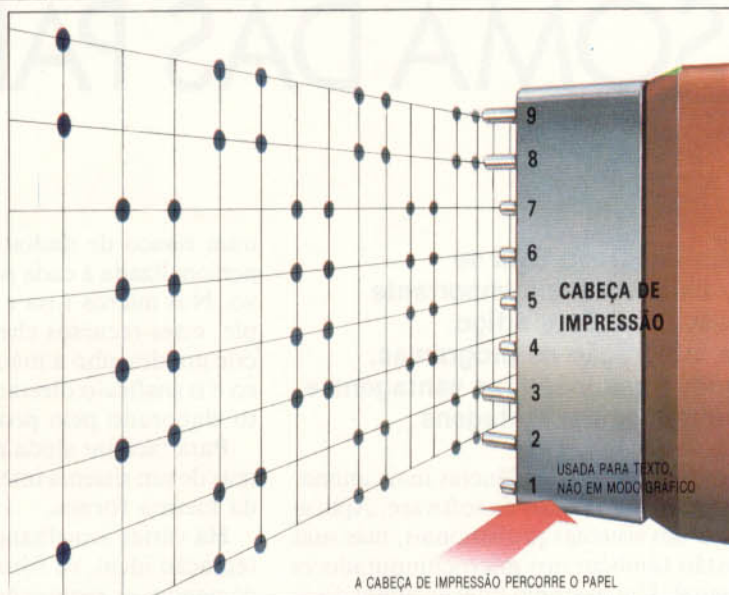
No modo gráfico comum, a FX-80 pode imprimir no máximo 480 pontos por linha. Outros modos permitem resoluções que vão de 576 a 1.920 pontos por linha. Portanto, para usar a largura toda, o comprimento da linha deve ser de 480. No nosso código, são necessários dois números para determinar a largura, pois o tamanho máximo de cada número é 255. O segundo número (N2) é, portanto, multiplicado por 256 e acrescentado ao primeiro (N1). Assim, para 480, os números são 1 e 224 ($480 = 256 \times 1 + 224$). Portanto, na impressora Epson FX-80 precisa-se da instrução:

```
LPRINT CHR$(27);"K";CHR$(224);CHR$(1);
```

Depois de programar a impressora com o comprimento de linha gráfica, é necessário enviar os dados do gráfico. Mesmo havendo nove agulhas na cabeça de impressão de uma Epson FX-80, só as oito de cima podem ser usadas na maioria dos modos gráficos. Começando pela agulha de baixo, vamos numerá-las: 1, 2, 4, 8, 16, 32, 64 e 128. Os dados para as oito agulhas podem então ser representados por um único número, entre 0 e 255, e isso é enviado à impressora usando-se LPRINT CHR$(X), onde X é o número. Assim, para ativar apenas a agulha de baixo, bas-

Pontilhado
O desenho foi produzido numa impressora do tipo matricial, enviando-se pares alternados dos números decimais 195 e 60 para a cabeça de impressão. O esquema *abaixo* mostra como estes números binários são interpretados pelas agulhas da cabeça de impressão *(ilustração à direita)*. A alimentação controlada do papel faz com que a linha seguinte cubra o espaço deixado pela agulha 1.

NÚMERO DA AGULHA	VALOR BINÁRIO		
9	128	•	○
8	64	•	○
7	32	○	•
6	16	○	•
5	8	○	•
4	4	○	•
3	2	•	○
2	1	•	○
		195	60

• AGULHA AVANÇA
○ AGULHA NÃO AVANÇA

ta mandar a instrução CHR$(1) para a impressora; para acionar somente a de cima, envia-se CHR$(128). Para uma combinação de agulhas, somam-se os números de cada uma. Esse processo é então repetido para cada um dos 480 pontos da linha.

Na ilustração, há dois padrões de agulhas: CHR$(195) e CHR$(60). Para imprimir as quatro primeiras colunas do padrão da linha, digita-se:

```
LPRINT CHR$(195);CHR$(195);CHR$(60);
   CHR$(60);
```

Após quatro colunas, o padrão se repete, e um loop FOR-NEXT elabora o resto da linha.

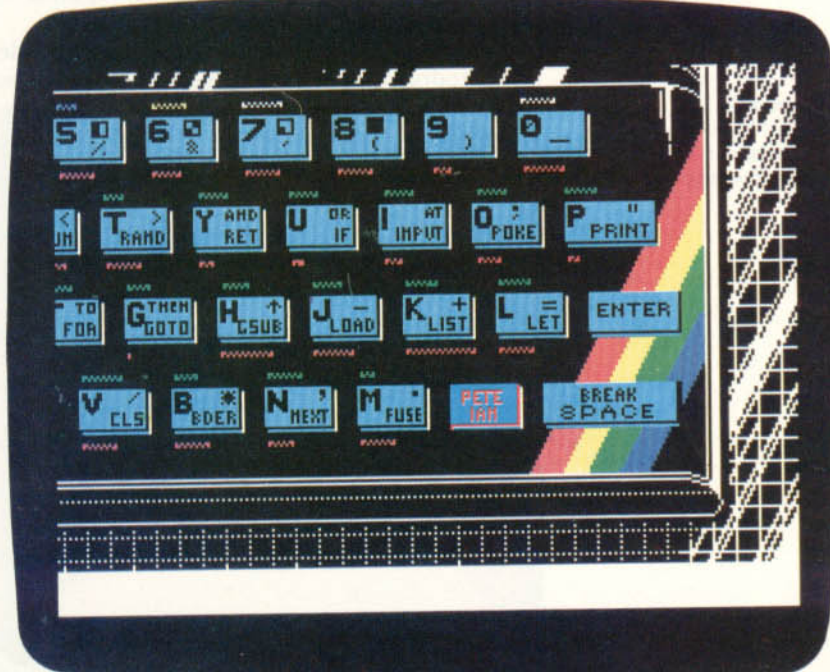
No exemplo, CHR$(60) não instrui a impressora a imprimir o caractere ASCII com o código 60 — é uma forma de representar os dados para as agulhas na cabeça de impressão. A impressora o reconhece como tal porque anteriormente foi transmitida a seqüência CHR$(27);"K" para acionar o modo gráfico.

Esse método de impressão por segmentos de imagem é descrito para uma impressora Epson FX-80; outras impressoras usam métodos semelhantes, com pequenas variações. Bastante trabalhosa, a produção de gráficos dessa maneira só é adequada quando há repetição de determinados padrões. Uma maneira mais eficiente de imprimir gráficos utiliza o dump de tela, um programa que copia no papel o que é apresentado no monitor.

Verificando toda a tela no sentido vertical e horizontal, o programa testa se o pixel está ligado em cada posição. Se estiver, deseja-se, então, que uma agulha da cabeça de impressão seja acionada na posição correspondente no papel. Para tanto, usa-se a função POINT(x,y), ou comandos semelhantes existentes na maioria dos micros; se um pixel estiver aceso, então a função POINT(x,y) será 1; se estiver apagado, a função será 0. As diferentes resoluções de tela de micros distintos significam que poderiam ser necessárias algumas adaptações.

Para um programa de dump de tela trabalhar com imagens coloridas, costuma-se atribuir diferentes padrões de pontos para cada cor. Um pixel de tela preto, por exemplo, imprime-se com quatro pontos em forma de quadrado; um vermelho pode ser representado por dois pontos; e um branco, por nenhum. A função POINT(x,y) produz um número diferente, conforme a cor do pixel, e isso também pode ser utilizado.

Os programas de dump de tela são, em geral, acrescentados no final do programa que produz a imagem, em forma de sub-rotinas. Para reproduzir a imagem na impressora, aperta-se a tecla [P], e o programa passa para a sub-rotina. Um programa de dump de tela escrito em BASIC tende a ser lento — leva até 5 min para imprimir uma figura pequena. As versões em código de máquina são ligeiramente mais rápidas.

Colorido
Esta imagem do teclado do Spectrum foi produzida numa impressora que funciona com jatos de tinta colorida — na prática, ela é uma impressora matricial, com jatos de tinta substituindo as agulhas.

A SOMA DAS PARTES

O software integrado está se tornando cada vez mais importante para os usuários. Este artigo analisa a integração de programas, comentando suas inúmeras vantagens e também algumas desvantagens.

A integração é uma das tendências mais animadoras já surgidas em termos de software. Aplica-se sobretudo aos sistemas profissionais, mas suas técnicas estão também nos microcomputadores de uso pessoal. Um exemplo internacional disso é o Sinclair QL, que possui quatro pacotes de software desenvolvidos de acordo com os princípios básicos da integração.

A principal conquista da integração é permitir que o usuário passe de um aplicativo para outro simples e rapidamente. Num sistema ideal, não haveria necessidade de encerrar um programa, retornar ao sistema operacional, trocar os disquetes e só então iniciar outro programa. Para ser eficaz, a troca de aplicativos tem de ocorrer a um toque no teclado, coisa que alguns programas, como o Lotus 1-2-3, o Symphony e o Framework, são capazes de fazer.

A possibilidade de transferir os dados facilmente entre os aplicativos também se mostra muito útil, por exemplo, para criar na planilha eletrônica uma coluna com os números referentes às vendas anuais de sua empresa e então transferi-la para o processador de texto, onde se escreverá um relatório anual.

Você pode usar, por meio do processador de texto, todos os nomes e endereços armazenados num banco de dados para escrever uma carta personalizada a cada pessoa registrada no arquivo. Nos micros Lisa e Macintosh, da linha Apple, esses recursos chegam a permitir que você crie um desenho a mão livre no programa gráfico e transfira-o diretamente para um documento elaborado pelo processador de texto.

Para facilitar ainda mais as tarefas, os programas de um sistema integrado em geral trabalham da mesma forma.

Há várias semelhanças em seu uso. Numa integração ideal, os formatos de tela, as teclas de comando, as requisições de dados, as mensagens de erro — enfim, todos os diferentes aspectos da interação com o usuário — devem ser idênticos ou, pelo menos, compatíveis. Caso contrário, o usuário não passa de um aplicativo para outro com facilidade.

A integração não seria eficaz se você tivesse de aprender a usar, digamos, cinco aplicativos com comandos de formatos diferentes. Mas se eles trabalharem da mesma forma, você só precisará aprender a usar um. Essa característica, conhecida como "padronização", vincula-se ao conceito de software integrado.

Lucros e perdas

Em síntese, o software integrado envolve três princípios básicos: a facilidade de passar de uma aplicação para outra, a possibilidade de permutar dados entre os programas e a padronização dos formatos. A integração contribui para tornar o computador mais acessível ao usuário médio, cujas necessidades são satisfeitas por dois ou três aplicativos. A integração tem aumentado a popularidade dos microcomputadores, que, graças a ela, se tornam mais eficientes e de uso menos complicado.

No entanto, os softwares integrados também têm suas desvantagens. Uma das principais é a exigência de grandes quantidades de RAM para operar. Às vezes é preciso colocar uma planilha, um processador de texto e um gerenciador de banco de dados — as três aplicações integradas com mais freqüência — em 16 ou 32 Kbytes.

Essa integração é possível, mas talvez não sobre espaço para o armazenamento dos dados. Tal problema restringe o uso dos softwares integrados às máquinas de maior capacidade: computadores com memória de 128 Kbytes ou mais. Os programas podem compartilhar algumas rotinas, que assim só precisam ser escritas uma vez. Contudo, cada aplicação tem suas próprias necessidades, que ocupam espaço na RAM.

Outra desvantagem: para economizar o espaço de memória requerido pelo programa, os pro-

Para avaliação
O Lotus 1-2-3, o Framework, o Open Access e o Symphony são bons exemplos de pacotes integrados.

gramadores de software precisam reduzir os recursos dos aplicativos. Um processador de texto integrado num pacote quase nunca é tão completo quanto um processador que opera sozinho, principalmente se este ocupa tanto espaço de memória quanto um pacote integrado.

Um bom exemplo disso são dois programas que rodam no IBM PC e em micros compatíveis. Um deles, o Multimate foi projetado a partir de software utilizado para processamento de textos: tem várias opções para criar e formatar textos, que não podem ser encontradas em programas menores e que tornam relativamente simples a elaboração de documentos longos e complexos. Sozinho, o Multimate exige 192 bytes para operar.

O outro é o Lotus 1-2-3, software que integra processador de texto, planilha e gerenciador de banco de dados e também ocupa 192 bytes. Ou seja: o processador de texto do Lotus tem apenas funções básicas, quando comparado ao Multimate.

Há uma terceira desvantagem na integração. É importante que os programas integrados se pareçam, para que seu uso seja fácil de aprender. Mas isso só é possível quando os programadores fazem algumas concessões. A melhor forma de operar uma planilha pode não ser a melhor para se usar os outros aplicativos. Os programadores precisam, então, combinar de modo eficiente os recursos mais apropriados a todos os programas.

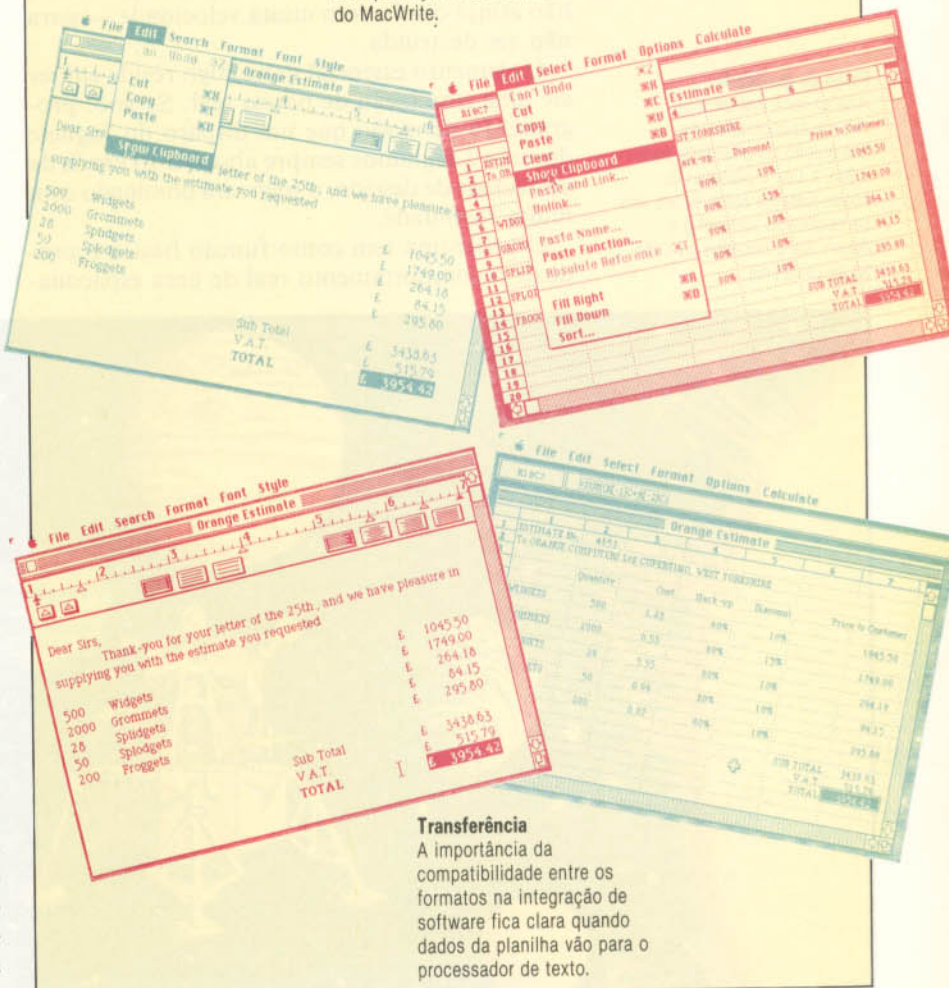
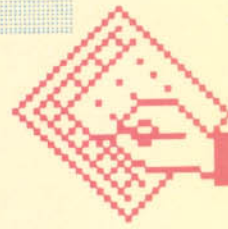
A Microsoft deparou-se com esse problema no projeto do processador de textos Word. A empresa queria que suas telas e sua operação fossem compatíveis com a planilha Multiplan (seu maior sucesso comercial), para que ambos pudessem ser facilmente integrados. Assim, a Microsoft incluiu no Word a mesma tela de menu que os usuários da planilha Multiplan haviam achado prática; só que não demorou muito para se chegar a esta conclusão: quem precisa de um programa como o Word não gosta de ter um menu na tela o tempo todo.

No lar e no trabalho

O ponto mais importante a ser lembrado em relação a softwares em geral é que eles devem fazer aquilo que o usuário quer. Se uma pessoa precisa executar várias tarefas (como cartas, mala direta e cálculos simples), o software integrado torna o trabalho muito mais fácil. Mas, caso o usuário pretenda escrever um romance ou elaborar relatórios empresariais muito longos, deve continuar utilizando programas separados, ou seja, processadores de texto, planilhas e gerenciadores de bancos de dados independentes.

Apesar disso, à medida que os programadores de software forem conseguindo comprimir instruções de programação em espaços cada vez menores, e a capacidade de memória dos equipamentos for se ampliando, o software integrado vai se tornar cada vez mais importante para os usuários, em casa e nos negócios.

Regras do jogo

Jogo combinado
Integração é a essência do micro Macintosh, que produziu estas ilustrações. A planilha eletrônica Multiplan, o processador de textos MacWrite e o gerador de gráficos MacPaint trocam dados por meio do sistema operacional, de modo que os três aplicativos, ao serem usados, parecem um só. Até na formatação de telas os pacotes do Macintosh são padronizados, como mostram as reproduções do Multiplan e do MacWrite.

Transferência
A importância da compatibilidade entre os formatos na integração de software fica clara quando dados da planilha vão para o processador de texto.

O SUPERVISOR

Sistema operacional ou **operating system:** software que controla e supervisiona todas as operações internas de um computador.

O POUSO DO MÓDULO

Este é um dos mais antigos jogos para computadores, e pode parecer simples demais diante de um do tipo fliperama. No entanto, quando bem programado, exige muita atenção e fica difícil de ser dominado.

O desafio não é simples: você é o piloto de um módulo lunar, o computador de bordo está em pane e o combustível começa a se esgotar. Cabe a você controlar o pouso, cuidadosamente, por meio de pequenos impulsos do foguete, de modo que o combustível não se esgote e que a nave não atinja o solo com muita velocidade — para não ser destruída.

O elemento essencial desse jogo reside em ser ele uma simulação de pouso real. Se você programá-lo de modo que um disparo do foguete durante 2 segundos sempre absorva 10 km/h da velocidade de descida, o jogo será dominado com muita facilidade.

O programa tem como função básica reproduzir o comportamento real de uma espaçonave, em condições tão precisas quanto possível. A matemática necessária para isso envolve cálculos muito complicados; portanto, o programa apresentado aqui é uma versão simplificada. Ainda assim, ele tem muitas das características de uma simulação autêntica.

Vamos ver em detalhes as condições para o pouso:

- O planeta no qual você está descendo tem determinada força de gravidade. Isso fará com que a espaçonave ganhe velocidade à medida que se aproxime dele.
- A espaçonave tem foguetes para contrabalançar os efeitos da gravidade empurrando o veículo no sentido contrário.
- A espaçonave é dotada de massa (e, sob gravidade, tem peso). Quanto maior essa massa, menor o efeito dos foguetes. Ao peso da espaçonave deve-se acrescentar o do combustível que ela transporta. À medida que o combustível é consumido, o módulo vai ficando mais leve.

Força da gravidade
Abaixo você encontra os valores aproximados da aceleração da gravidade para o Sol, a Lua e planetas de nosso sistema solar. Esses são os valores da variável g e devem ser colocados na linha 30 do programa.

Sol -274
Júpiter -25
Saturno -10,4
Plutão -4
Urano -10,4
Vênus -8,8
Marte -3,8
Terra -9,8
Mercúrio -3,7
Lua -1,6

Assim, para traduzir matematicamente a maneira pela qual nosso veículo se comporta, precisamos de um conjunto de equações que envolvem aceleração, massa, velocidade etc.

Se elas vão ser muito simples ou muito complicadas, depende do grau de precisão e do número de pormenores que desejamos levar em conta. Mas, para os propósitos de nosso jogo, vamos manter esses fatores num nível razoavelmente simples.

A informação mais importante de que precisamos é a altitude em que a nave se encontra. Ela se move continuamente, seja caindo sob efeito da gravidade, seja indo para longe do planeta devido ao uso exagerado dos foguetes. Para que possamos saber sua posição em determinado instante, dividimos o tempo numa série de passos — ou períodos.

Em cada período, podemos calcular o espaço percorrido pelo módulo, as mudanças no valor de sua velocidade, seu peso etc. Esses períodos podem ter a duração que você quiser — quanto menores, mais precisa a simulação.

A velocidade é medida em unidade por hora, o que se reduz à fórmula:

$$\text{distância} = \text{tempo} \times \text{velocidade}$$

Podemos, assim, calcular a distância que o módulo percorre, para cima ou para baixo, multiplicando sua velocidade pela duração do período (que definimos como unidade). Dessa forma, ajustamos a velocidade do módulo acelerando-o com a ajuda da gravidade do planeta e desacelerando-o por meio dos foguetes.

A aceleração da gravidade, sempre constante, corresponde à variável g, no programa. Ela difere em cada planeta onde você irá pousar. A ilustração mostra os valores de g para os planetas de nosso sistema solar, mas você poderá experimentar outros valores, ou fazer o programa gerar para g um valor aleatório, produzindo assim um jogo mais difícil.

Simular o motor do foguete é um pouco mais complicado. Na versão que apresentamos, o jogador consome de uma a nove unidades de combustível por período; o programa calculará a aceleração resultante, levando em conta a massa do veículo. Na vida real, a fórmula exata depende da potência do motor do foguete e do tipo de combustível utilizado. Em nosso programa, os valores foram escolhidos de maneira a tornar o jogo difícil; no entanto, você pode tentar alterá-los para ver como eles afetam o resultado.

Um detalhe acrescentável é a possibilidade de inserir novos dados em tempo real, ou seja, sem que o programa pare de rodar. Essa é, muitas vezes, a única diferença que transparece entre o uso de uma instrução INPUT (que interrompe a execução do programa a fim de reunir informações que o usuário forneça) e uma instrução INKEY$ ou GET.

O jogo ficará melhor se não tiver de ser interrompido para calcular quanto combustível deve ser queimado. Caso isso aconteça, o jogador terá tempo para examinar a situação e fazer alguns cálculos — ao passo que, na vida real, isso exigiria raciocínio muito rápido.

Nesse programa de pouso, um loop é executado todas as vezes que se alcança a linha 250. Ajustando-se o período de modo a igualar o tempo gasto na execução do loop, a simulação vai funcionar em tempo real: a alunissagem, na simulação, vai durar tanto quanto uma real.

Embora numa simulação isso seja desejável, pode ser muito difícil de conseguir num jogo como este. Em geral, a simulação é muito demorada para apresentar interesse como jogo.

Elementos extras

Existem muitas coisas que você pode fazer em seu programa básico de alunissagem. A mais óbvia delas consiste em acrescentar alguns detalhes gráficos para descida.

As idéias para colocar isso em prática variam desde um simples altímetro com o mostrador redondo até a paisagem do local de pouso, vista de cima e graduada por uma escala. Você também pode acrescentar movimentos laterais, embora o módulo, normalmente, não se desloque nessa direção.

Porém, se estiver pensando num planeta com atmosfera, você poderá acrescentar, como uma dificuldade a mais, os efeitos de um vento na superfície ou coisa semelhante.

Algumas versões sofisticadas do programa possuem várias áreas de pouso, situadas no fundo de túneis e de crateras, o que exige uma pilotagem cuidadosa.

Alunissagem pelo vídeo

```
  10 REM PARA A LINHA SINCLAIR
  20 REM       POUSO NA LUA
  30 LET G=-1.6
  40 LET T=1
  50 LET C=1000
  60 LET V=0
  70 LET H=2000
  80 LET M=2000+C
  90 LET G=G+T
 100 REM ATUALIZA TELA
 110 PRINT AT 0,9;"POUSO NA LUA"
 120 PRINT AT 2,0;"ALTURA....:";
INT H;"  "
 130 PRINT ,,"VELOCIDADE....:";I
NT V;"  "
 140 PRINT ,,"COMBUSTIVEL....:";
INT C;"  ",,
 150 IF H<0 THEN GOTO 310
 160 IF C>0 THEN GOTO 190
 170 PRINT ,,"SEM COMBUSTIVEL
"
 180 GOTO 200
 190 PRINT ,,"POTENCIA DO MOTOR
0-9"
 200 LET B=0
 210 IF C<0 THEN GOTO 240
 220 LET A$=INKEY$
 230 IF A$>="0" AND A$<="9" THEN
 LET B=VAL A$
 240 IF B>C THEN LET B=0
 250 LET H=H+V*T
 260 LET V=V+G
 270 LET V=V+(B*3000)/M
 280 LET C=C-B
 290 LET M=M-B
 300 GOTO 100
 310 REM NA SUPERFICIE DA LUA
 320 IF V>-10 THEN PRINT AT 10,0
;"POUSO PERFEITO...PARABENS"
 330 IF V<=-10 AND V>-20 THEN PR
INT AT 10,0;"CRASH...VOCE ARREBE
NTOU A NAVE  MAS OS TRIPULANTES
SOBREVIVERAM"
 340 IF V>-20 THEN GOTO 370
 350 PRINT AT 10,0;"CRUNCH...ESP
ACONAVE DESTRUIDA...SEM SOBREVIV
ENTES"
 360 PRINT "VOCE ACABA DE CRIAR
UMA NOVA CRATERA DE ";INT (-V*.1
);" KM DE DIAMETRO"
 370 PRINT ,,"JOGA NOVAMENTE ?(S
/N)"
 380 LET A$=INKEY$
 390 IF A$="" THEN GOTO 380
 400 IF A$="S" THEN RUN
 410 IF A$<>"N" THEN GOTO 380
 420 STOP
```

Este programa roda em TK 85, TK 90X e CP 200.

I-7000 PCxt

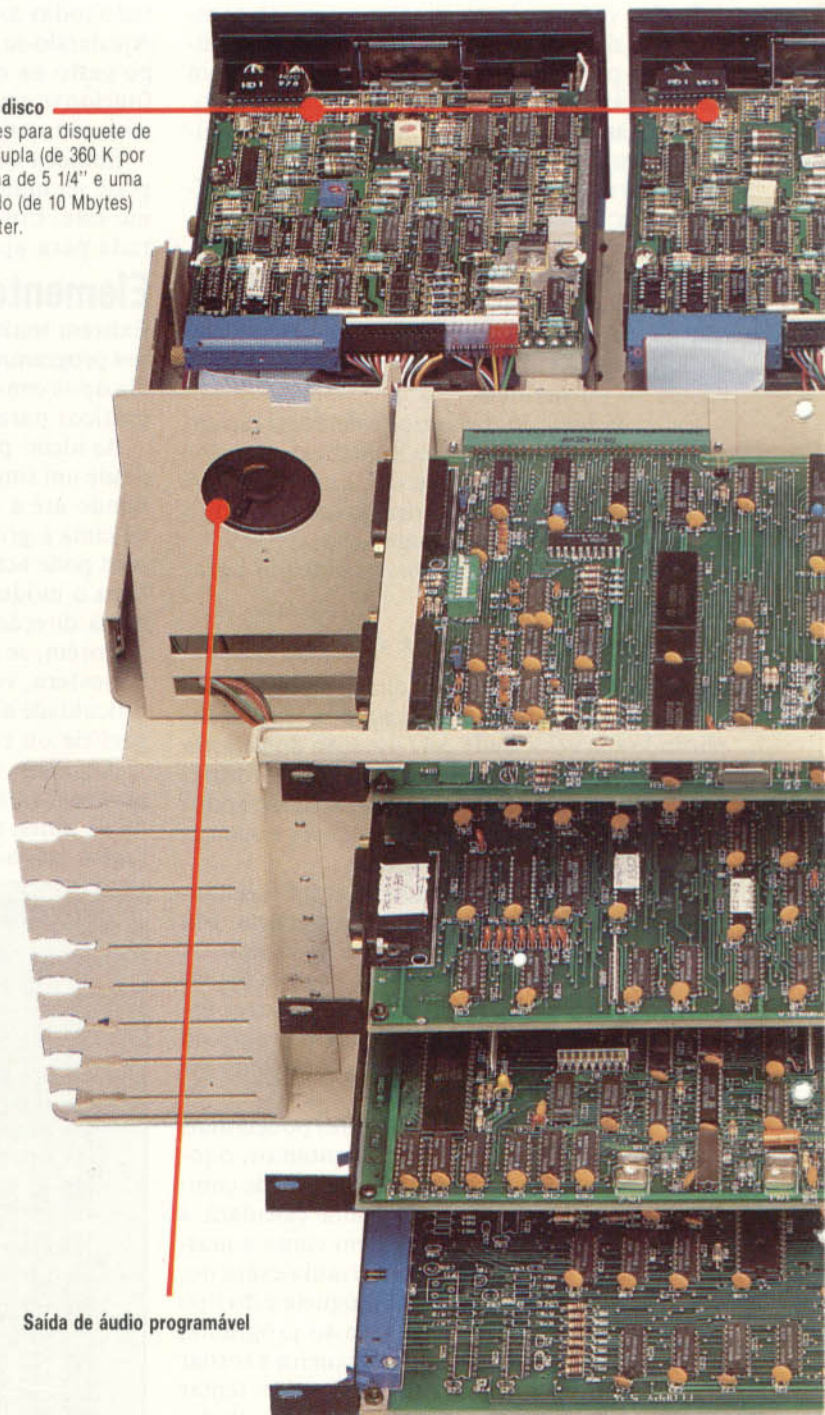
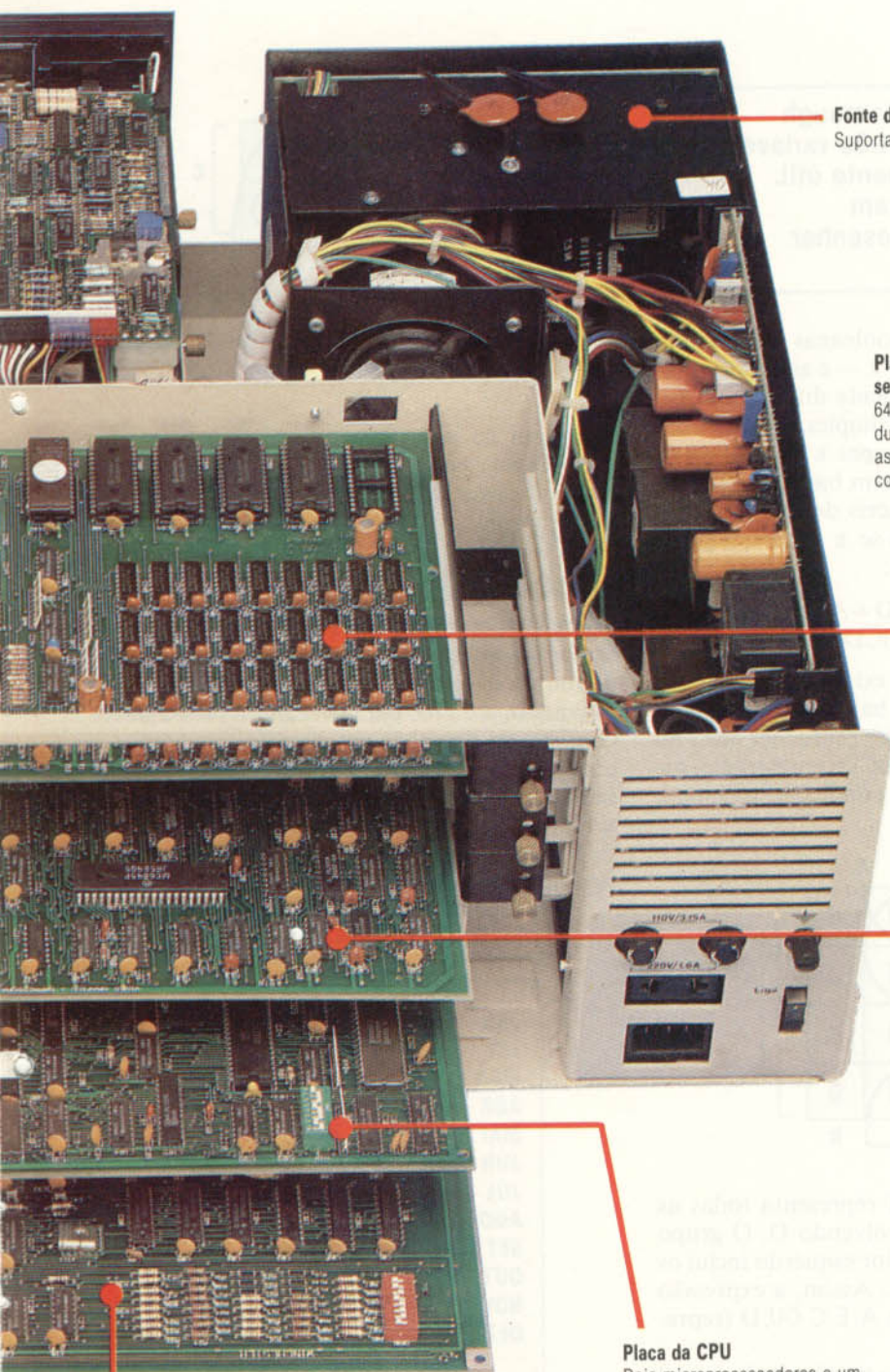
Incorporando diversos recursos em sua moderna arquitetura de 16 bits, o I-7000 PCxt, da Itautec, oferece alto desempenho e compatibilidade com a maior parte dos aplicativos e linguagens de programação.

Equipado com memória de 256 Kbytes, expansível até 640 Kbytes, duas CPUs de alta velocidade e dois processadores auxiliares, o I-7000 PCxt deve à sua arquitetura de 16 bits o elevado desempenho que o caracteriza. Capaz de processar grande volume de dados em curto espaço de tempo, esse equipamento aceita qualquer programa, periférico ou acessório, compatível com o IBM PC.

Sua alta resolução gráfica, de 640 x 400 pixels endereçáveis na tela, permite criar figuras com grande riqueza de detalhes. Utilizando monitor em cores, trabalha com até dezesseis cores ao mesmo tempo; e, na resolução gráfica máxima, a máquina pode gerar telas compostas de até quatro cores.

O I-7000 PCxt opera automaticamente com os sistemas SIM/M e SIM/DOS, compatíveis com o CP/M e o MS/DOS, recurso responsável pela versatilidade do equipamento: pode ser utilizado com diversos aplicativos e linguagens de programação, além de rodar também programas desenvolvidos para o I-7000 e o I-7000 Jr., da Itautec. Graças às interfaces de comunicação, o I-7000 PCxt tem acesso a computadores de grande porte — IBM, DEC, Burroughs —, o que viabiliza a criação de redes de micros e conexão a sistemas de comunicação de dados como, por exemplo, Cirandão e Videotexto.

O teclado projetado para a língua portuguesa tem inclinação ajustável, é independente e dis-

Unidades de disco
Duas unidades para disquete de 5 1/4", face dupla (de 360 K por disco), ou uma de 5 1/4" e uma de disco rígido (de 10 Mbytes) tipo Winchester.

Saída de áudio programável

Fonte de alimentação
Suporta disco rígido.

Placa de memória e interfaces seriais
640 K de RAM e 48 K de ROM; duas interfaces seriais assíncronas RS232C ou loop de corrente.

Placa de vídeo e impressora
Interface paralela padrão Centronics. Controladora de vídeo colorido (dezesseis cores) com capacidade gráfica de 640 x 400 pixels.

Placa da CPU
Dois microprocessadores e um co-processador numérico.

Placa controladora de disquete 5 1/4" e Winchester
Controla quatro disquetes e o disco rígido.

I-7000 PCxt

MICROPROCESSADORES

CPUs: 8088-2 e Z80B. Auxiliares: 8035 (controlado por teclado) 8087-2 (aritmético de ponto flutuante — opcional).

CLOCK

8088-2: 4,77 ou 8 MHz, selecionáveis por software
Z80B: 6 MHz

MEMÓRIA

256 K de RAM, expansíveis até 640 K;
48 K de ROM.

VÍDEO

Monitor de 12", fósforo verde (oito tons).

SISTEMA OPERACIONAL

SIM/M, compatível com CP/M.
SIM/DOS, compatível com MS/DOS.

TECLADO

99 teclas (com todos os caracteres da língua portuguesa), sendo doze de funções programáveis e quatro para controle do cursor. Bloco numérico separado.

LINGUAGENS

BASIC, COBOL, LOGO e outras que rodem sob CP/M ou MS/DOS.

INTERFACES

Saída SASI para disco rígido; saída paralela padrão Centronics; duas saídas seriais padrão RS232C.

DOCUMENTAÇÃO

Manual do usuário, completo e bem ilustrado.

põe de 99 teclas tipo máquina de escrever, doze das quais programáveis e quatro destinadas ao controle do cursor. Há também um bloco numérico em separado.

O equipamento compõe-se de três módulos independentes: módulo básico, teclado e monitor de vídeo. A montagem "em torre" do módulo básico permite colocá-lo no solo, em posição vertical, o que resulta em melhor aproveitamento do espaço, pois libera a mesa do operador.

O I-7000 PCxt conta com duas unidades para disquete de 5 1/4 polegadas, face dupla, ou uma de 5 1/4 polegadas e uma de disco rígido. Todos os elementos da máquina estão contidos em quatro placas de circuito. Há ainda três slots livres para placas de expansão.

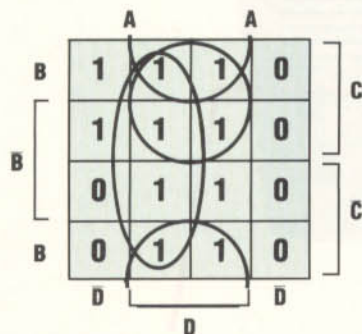
MAPAS E CIRCUITOS

O uso dos mapas de Karnaugh nos casos de mais de três variáveis costuma ser extremamente útil. E os mapas-k simplificam muito o processo de desenhar circuitos eletrônicos.

No caso de expressões booleanas envolvendo quatro variáveis, os mapas-k — e as próprias expressões — são aparentemente difíceis. No entanto, aplicando-se noções simples, estabelecidas quando examinamos os mapas-k de duas e três variáveis, eles logo se tornam bastante familiares e, mais ainda, bem fáceis de manipular.

Por exemplo, suponha-se a necessidade de simplificar esta expressão:

$$ABC\overline{D}+ABCD+\overline{A}BCD+A\overline{B}C\overline{D}+\overline{B}CD+$$
$$+A\overline{B}\overline{C}D+\overline{A}\overline{B}\overline{C}D+B\overline{C}D$$

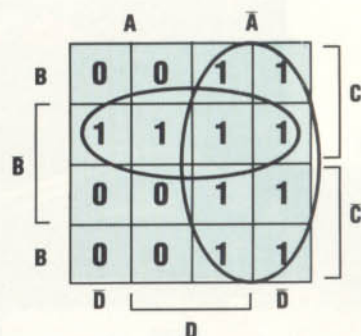
Pode-se perceber que ela exige um mapa-k de quatro variáveis. Embora haja oito componentes na expressão, será preciso preencher dez 1 no mapa-k (os termos $\overline{B}CD$ e $B\overline{C}D$ representam, cada um, dois casos). Nessas condições, o mapa-k assim se apresenta:

O grupo central de oito 1 representa todas as combinações possíveis envolvendo D. O grupo de quatro 1 no canto superior esquerdo inclui os casos que envolvem A E C. Assim, a expressão pode ser simplificada para A E C OU D (representada como $A.C+D$).

Às vezes é necessária uma manipulação inicial da expressão para dispô-la numa forma adequada para a representação num mapa-k, como no seguinte exemplo:

$$\overline{A+B+C}+\overline{A}.B+\overline{B+\overline{C}}$$

Aqui devemos aplicar a lei de De Morgan à expressão, antes que seja possível obter o mapa-k. Disso resulta:

$$\overline{A}.\overline{B}.\overline{C}+\overline{A}.B+\overline{B}.C$$

e o mapa-k que essa expressão produz é:

Por meio do mapa-k, essa expressão pode ser simplificada para: $\overline{A}+\overline{B}.C$. Novamente pela lei de De Morgan, simplifica-se para:

$$\overline{A.(B+\overline{C})}$$

Desenho do circuito

Exemplo 1: meses de trinta dias

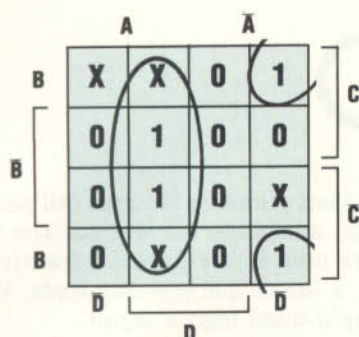
Suponhamos que cada mês do ano seja indicado num código binário de 4 bits, desde 0001 para janeiro, até 1100 para dezembro. Nossa tarefa consiste em desenhar um circuito que admita o código de 4 bits como entrada e produza uma saída de 1 se o mês que for introduzido tiver trinta dias.

A tabela de validação para um circuito desse tipo será:

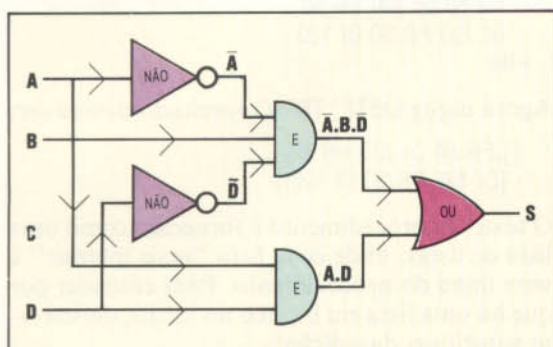
MÊS	ENTRADA				SAÍDA
	A	B	C	D	S
	0	0	0	0	X
JAN	0	0	0	1	0
FEV	0	0	1	0	0
MAR	0	0	1	1	0
ABR	0	1	0	0	1
MAI	0	1	0	1	0
JUN	0	1	1	0	1
JUL	0	1	1	1	0
AGO	1	0	0	0	0
SET	1	0	0	1	1
OUT	1	0	1	0	0
NOV	1	0	1	1	1
DEZ	1	1	0	0	0
	1	1	0	1	X
	1	1	1	0	X
	1	1	1	1	X

A saída X da tabela de validação significa uma entrada não válida. Vamos supor que o circuito *não* vá receber tais sinais. Pela tabela de validação e sempre que $S=1$, podemos formar a seguinte expressão booleana com os bits binários da entrada:

$$S=\overline{A}.B.\overline{C}.\overline{D}+\overline{A}.B.C.\overline{D}+A.\overline{B}.\overline{C}.D+$$
$$+A.\overline{B}.C.D$$

Levando essa expressão a um mapa-k, juntamente com as condições (X) de "entrada não válida", obtemos:

A partir desse mapa-k, podemos ver que a expressão se reduz a:

$A.D+\overline{A}.B.\overline{D}$

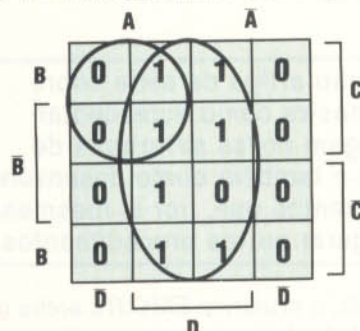
E assim, nosso circuito de sinal "mês de trinta dias" pode ser construído:

Exemplo 2: números ímpares

Considerando que os números de 0 a 15 podem ser codificados por quatro dígitos binários (0000 a 1111), desenhar um circuito que admita o código de 4 bits como uma entrada e produza uma saída de 1, sendo a saída um número ímpar maior que 2. A primeira providência é construir uma tabela de validação para todas as condições:

NÚMERO DECIMAL	ENTRADAS A	B	C	D	SAÍDA S
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1

Com essa tabela de validação, formamos a seguinte expressão da álgebra booleana, para todas as condições em que S é verdadeira (=1):

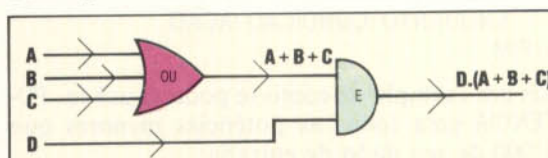
$$S=\overline{A}.\overline{B}.C.D+\overline{A}.B.\overline{C}.D+\overline{A}.B.C.D+$$
$$+A.\overline{B}.\overline{C}.D+A.\overline{B}.C.D+A.B.\overline{C}.D+$$
$$+A.B.C.D$$

O mapa de Karnaugh para essa expressão é:

A partir do mapa-k, três grupos de 4 podem ser isolados segundo a expressão

$S=A.D+C.D+B.D$

Isso pode ser mais simplificado ainda, por meio da lei distributiva, para obter:

$S=D.(A+B+C)$

Conseqüentemente, o circuito resultante é:

O próximo artigo da série contém uma revisão dos aspectos mais importantes da lógica booleana e apresenta exercícios de revisão.

Exercício 5

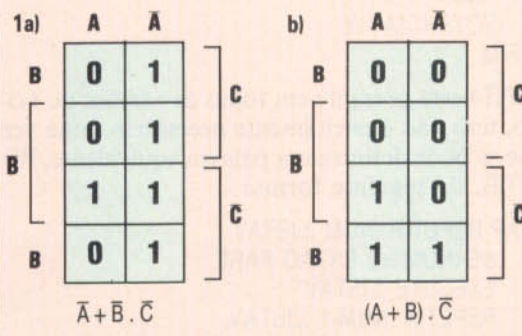
1) Simplificar as seguintes expressões booleanas, utilizando os mapas de Karnaugh:

a) $A.B.C+A.\overline{B}.\overline{C}+\overline{A}.\overline{C}+\overline{A}.B.C+A.B.\overline{C}$

b) $\overline{B+C}+B.\overline{C}+A.C$

c) $A.\overline{B}.D+\overline{A}.D+A.B.C.D+A.B.\overline{C}+\overline{A}.B.\overline{C}.\overline{D}$

2) Desenhar um circuito que admita as representações binárias dos números inteiros entre 0 e 7, inclusive. O circuito deverá resultar numa saída se o número introduzido for ímpar ou se for um múltiplo de 3 (isto é, 3 ou 6). A partir da tabela de validação e da expressão simplificada, desenhe um circuito lógico para essa função.

Respostas do exercício 4

REPITA O DESEMPENHO

Este último artigo da série sobre o LOGO mostra como acrescentar à linguagem novas estruturas de controle e também como desenvolver procedimentos que, por si mesmos, podem gerar outros procedimentos.

No MLOGO, o primitivo EXECUTE aceita uma lista como dado de entrada e a executa exatamente como se fosse uma linha de procedimento. Emprega-se esse recurso para o acréscimo — quando necessário — de novas estruturas de controle à linguagem. Assim, define-se o procedimento ENQUANTO da seguinte forma:

```
AP ENQUANTO :CONDICAO :ACAO
  SE NAO (EXECUTE :CONDICAO) ENTAO PARE
  EXECUTE :ACAO
  ENQUANTO :CONDICAO :ACAO
FIM
```

Eis um exemplo de como se poderia usá-lo. POTENCIA gera todas as potências menores que 1.000 de seu dado de entrada:

```
AP POTENCIA :X
  FACA "P :X
  ENQUANTO [:P<1000][MOSTRE :P
    FACA "P :P * :X]
FIM
```

Embora comuns em outras linguagens, as estruturas de controle — como ENQUANTO e REPITA — não são realmente necessárias em MLOGO. Um modo mais natural de escrever POTENCIA em MLOGO seria:

```
AP POTENCIA1 :P
  SE NAO :P < 1000 ENTAO PARE
  MOSTRE :P
  POTENCIA1 :P * :I
FIM
```

```
AP POTENCIA :X
  FACA "I :X
  POTENCIA1 :X
FIM
```

REPITA está presente em todas as versões do LOGO, mas não é estritamente necessário, uma vez que se pode definir uma palavra equivalente, REPETIR, da seguinte forma:

```
AP REPETIR :NUM :LISTAV
  SE :NUM=0 ENTAO PARE
  EXECUTE :LISTAV
  REPETIR :NUM-1 :LISTAV
FIM
```

EXECUTE é um primitivo bastante útil para o trabalho mais adiantado em MLOGO. Um programa monta uma lista e depois a transfere para EXECUTE, a fim de que seja executada. Veremos um exemplo disso logo a seguir.

Desmontando os procedimentos

Em primeiro lugar, definimos um procedimento para desenhar um triângulo da forma habitual:

```
AP TRI
  FR 50 DI 120 FR 50
  DI 120 FR 50 DI 120
FIM
```

Agora digite LISTE "TRI. O resultado deverá ser:

```
[[][FR 50 DI 120 FR 50]
  [DI 120 FR 50 DI 120]]
```

O texto do procedimento é fornecido como uma lista de listas, onde cada lista ''mais interior'' é uma linha do procedimento. Para entender por que há uma lista em branco no início, defina este substituto da adição:

```
AP SOMA :A :B
  MOSTRE :A+:B
FIM
```

Agora, obteremos, com LISTE "SOMA:

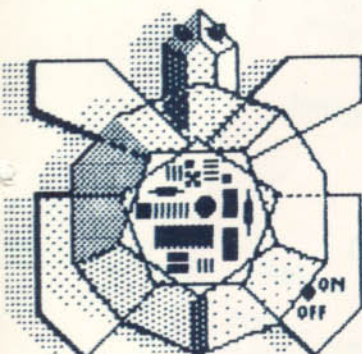
```
[:A :B][MOSTRE :A+:B]
```

A primeira lista contém os dados para o procedimento. Assim, LISTE permite verificar o que está contido num procedimento. DEFINA, pelo contrário, permite-nos definir um procedimento como uma lista de listas sem que seja preciso utilizar o editor. Tente agora DEFINA "L [[:A] [FR :A][DI 90][FR :A/2]] e então execute L usando, por exemplo, L 30. O uso de DEFINA em modo imediato não apresenta vantagens em relação ao uso do editor, mas apenas na capacidade de um procedimento gerar outro procedimento.

Crescimento

Vamos agora desenvolver um pequeno sistema para o crescimento de figuras. Seus comandos básicos são FAZER, que escolhe a forma com que iremos lidar, e CRESCER, que modifica o tamanho da forma escolhida. Por exemplo, FAZER "QUADRADO desenha um quadrado e então CRESCER [*10] o apaga, desenhando-o novamente com os lados aumentados por um fator 10.

Para simplificar os programas, aceitaremos algumas restrições quanto ao que podemos realizar com esses comandos. Os programas para

Desenhe uma tartaruga
Não se pode avançar muito no LOGO sem a recorrência: algo definido em termos de si mesmo, como os procedimentos que se invocam, as listas definidas por listas e os procedimentos para criar procedimentos. Com um pouco de imaginação, é possível fazer um desenho no estilo de Escher, usando uma tartaruga para gerar outra tartaruga que desenha uma terceira tartaruga...

desenhar formas — fornecidos a FAZER como entradas — não podem conter o comando REPITA ou chamada de outros programas. Em segundo lugar, o sistema não funciona com resultados negativos. Nenhum desses dois problemas será muito difícil de resolver se você quiser melhorar o que aqui apresentamos.

FAZER atribui o nome da forma à variável global "ATUAL e então executa o programa. Para isso, ele cria uma lista de um item — o nome do programa — e então usa EXECUTE para rodá-lo.

```
AP FAZER :OBJETO
  MOSTRET
  FACA "ATUAL :OBJETO
  EXECUTE (LISTA :OBJETO)
FIM
```

CRESCER primeiro elimina o desenho inicial e depois usa DEFINA para definir o procedimento atual como um procedimento reescrito. A cor retorna então ao normal e a nova forma é desenhada. Observe que a entrada para CRESCER é armazenada na variável OPLISTA — que teremos de usar depois.

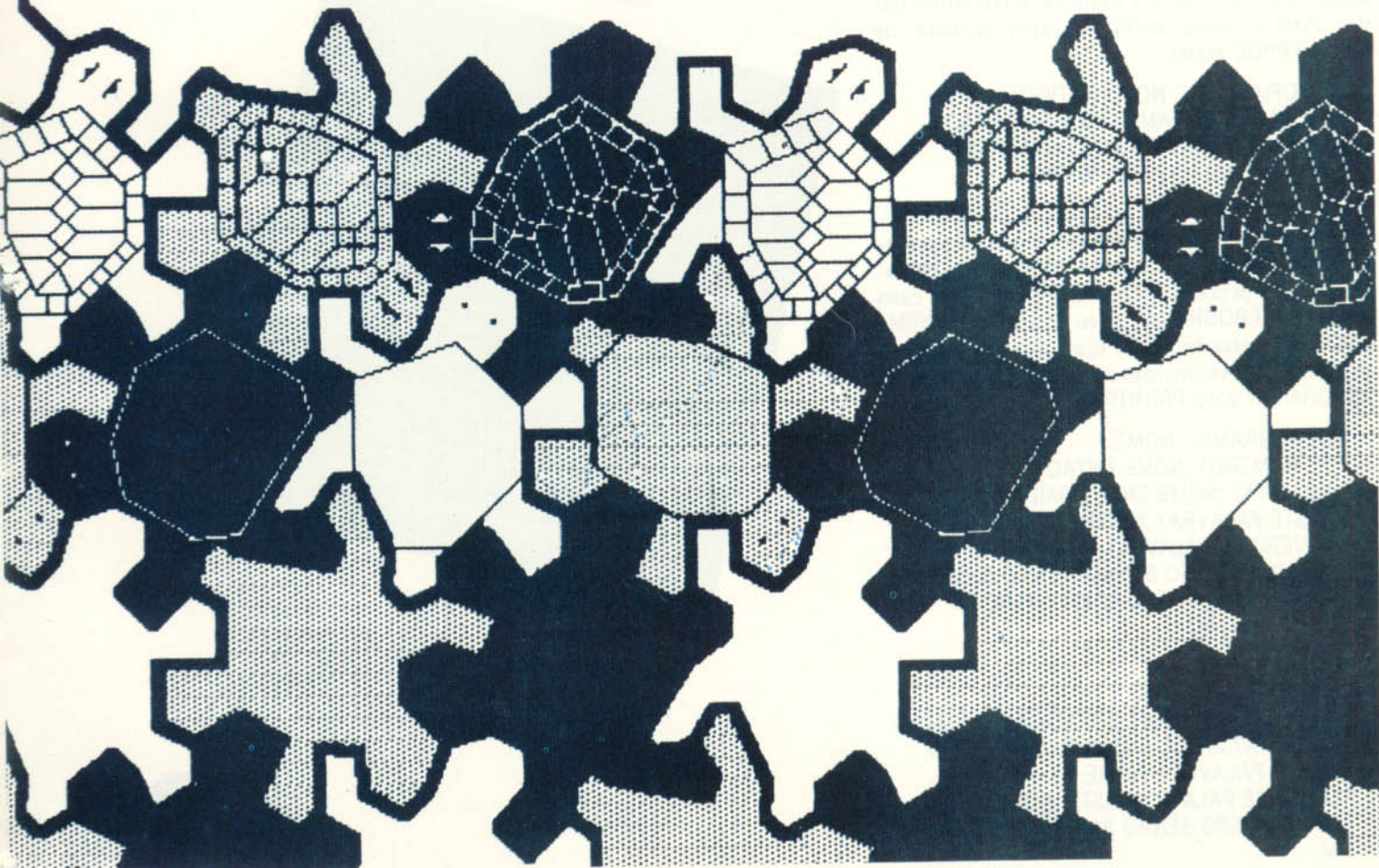
```
AP CRESCER :OPLISTA
  COR 0
  EXECUTE (LISTA :ATUAL)
  DEFINA "ATUAL ESCREVA.PROC LISTE
    :ATUAL
  COR 1
  EXECUTE (LISTA :ATUAL)
FIM
```

ESCREVA.PROC divide o texto em linhas e as transfere, uma por vez, para ESCREVA.LINHA:

```
AP ESCREVA.PROC :TEXTO
  SE :TEXTO = [] ENTAO SAIDA []
  SAIDA PENT ESCREVA.LINHA PRIMEIRO
    :TEXTO ESCREVA.PROC SEMPRIMEIRO
    :TEXTO
FIM
```

ESCREVA.LINHA examina uma linha, buscando um FR ou FRENTE. Se encontrar, ele passa o resto da linha para MUDE.

```
AP ESCREVA.LINHA. :LINHA
  SE :LINHA = [] ENTAO SAIDA []
  SE SEUM PRIMEIRO :LINHA = "FR
    PRIMEIRO :LINHA = "FRENTE ENTAO
    SAIDA MUDE SEMPRIMEIRO :LINHA
  SAIDA PENT PRIMEIRO :LINHA
    ESCREVA.LINHA SEMPRIMEIRO :LINHA
FIM
```

MUDE constrói a linha "reescrita". O primeiro

item de LISTAV — a entrada para MUDE — teria sido a entrada para FRENTE no procedimento inicial. Se ela for, por exemplo, 50 e se OPLISTA contiver [*2], então SENTENCA PRIMEIRO :LISTAV :OPLISTA será [50 * 2]. MUDE agora usa EXECUTE para calcular o valor dessa lista (obtendo o resultado 100). Finalmente, obtém-se uma lista que consiste em FR, na quantidade cujo valor acaba de ser calculado e, a seguir, na reescrita do resto da linha.

```
AP MUDE :LISTAV
  SAIDA (SENTENCA "FR (EXECUTE
    SENTENCA PRIMEIRO :LISTAV
    :OPLISTA) ESCREVA.LINHA
    SEMPRIMEIRO :LISTAV)
FIM
```

Cópias

Às vezes é útil fazer uma cópia de um programa. Assim, vamos definir um programa COPIAR.PROC — de modo que COPIAR.PROC "NOVO. .NOME "NOME.ANTIGO defina NOVO.NOME como uma cópia de NOME.ANTIGO (NOME.ANTIGO permanece não afetado). Uma definição evidente é a seguinte:

```
AP COPIAR.PROC NOVO ANTIGO
  DEFINA "NOVO LISTE "ANTIGO
FIM
```

O problema com essa definição está em que, se ANTIGO não existir, então o programa simplesmente seguirá adiante e definirá NOVO como nada. Assim, uma definição aperfeiçoada de COPIAR.PROC seria:

```
AP COPIAR.PROC NOVO ANTIGO
  SE NAO PROGRAMA? :ANTIGO ENTAO
    MOSTRE [NAO E PROGRAMA] PARE
  DEFINA "NOVO LISTE "ANTIGO
FIM
```

Essa definição utiliza um procedimento chamado PROGRAMA?, que resulta VERD se sua entrada é um procedimento, e FALSO em caso contrário. PROGRAMA? e seu equivalente PRIMITIVO? são testes bastante úteis que não existem em MLOGO. Assim, desenvolvemos versões de PROGRAMA? e de PRIMITIVO?:

```
AP PROGRAMA? :NOME
  SE NUMERO? :NOME ENTAO SAIDA "FALSO
  SE LISTA? :NOME ENTAO SAIDA "FALSO
  TESTE PALAVRA? :NOME
  SEVERD SE PALAVRA? LISTE :NOME ENTAO
    SAIDA "FALSO SENAO SE NAO (LISTE
    :NOME = []) ENTAO SAIDA "VERD
  SAIDA "FALSO
FIM

AP PRIMITIVO? :NOME
  SE NUMERO? :NOME ENTAO SAIDA "FALSO
  SE LISTA? :NOME ENTAO SAIDA "FALSO
  TESTE PALAVRA? :NOME
SEVERD SE PALAVRA? LISTE :NOME ENTAO
  SAIDA "VERD SENAO SAIDA "FALSO
FIM
```

FONTES DE ERROS

Enganos conceituais ou erros de lógica podem produzir resultados catastróficos. As partes potencialmente problemáticas são as ligações entre as rotinas e entre o programa e seu usuário.

OVERFLOW

Estouro, excesso. Quantidade que ultrapassa a capacidade de armazenamento de um registro.

Existem muitas fontes potenciais de erros a cada estágio da elaboração do programa, desde sua especificação até os testes, passando pelo projeto e codificação. Nos estágios de especificação e projeto, os erros costumam ocorrer quando não se verifica com precisão o tipo de problema a ser resolvido e não se toma cuidado suficiente para garantir que o programa faça exatamente o que foi determinado.

A probabilidade de erros diminui quando se segue o método de projeto estruturado. Outros erros ainda podem acontecer ao se transformar o projeto em instruções, na fase de digitação ou na de testes e eliminação de erros. A própria correção de um erro leva às vezes a outros.

Checagem de erros

Uma abordagem logicamente estruturada constitui a essência para evitar e eliminar erros; a lista de checagem de erros deste quadro é um exemplo abreviado de tal abordagem:

Variáveis

1 Todas as variáveis têm nomes exclusivos, levando-se em consideração que muitos interpretadores usam somente os dois primeiros caracteres de cada nome?
2 Alguma variável (sobretudo contadores de loop ou parâmetros de sub-rotinas) foi reutilizada enquanto seu conteúdo era ainda significante?
3 Os índices das matrizes estão dentro dos limites? São números inteiros?
4 Os índices das matrizes começam com o elemento 0 ou 1?

Cálculos

1 Os cálculos produzem resultados numéricos ou alfabéticos? E as variáveis para os resultados são alfabéticas ou numéricas?
2 Algum cálculo resulta num número pequeno ou grande demais para o computador trabalhar? Isso pode ocasionar um erro de divisão por zero?
3 Os erros de arredondamento de números podem ser significativos?
4 As operações numa expressão executam-se na ordem lógica correta ou na ordem imposta pela prioridade dos operadores aritméticos?

Comparações

1 Os strings são sempre comparados com strings, e os números com números?
2 Faz diferença se um string de teste é escrito em maiúsculas ou minúsculas?
3 Há comparações de strings de tamanhos diferentes?
4 Os operadores booleanos e os de comparação estão sendo utilizados de modo adequado? Por exemplo, A>B OU C não é o mesmo que A>B OU A>C. A prioridade dos operadores booleanos e dos de comparação afeta a execução das expressões de comparação?

Controle

1 Os loops e os algoritmos terminam qualquer que seja a situação das variáveis?
2 Os loops e as rotinas têm, cada um, somente um ponto de entrada e um ponto de saída?
3 Quando um comando IF-THEN falha, o controle passa para a próxima instrução ou para a próxima linha do programa?
4 O que acontece se nenhuma das condições de teste numa ramificação múltipla é satisfeita?

Mas é nas interfaces — entre rotinas e entre o programa e o usuário — que ocorre a maioria dos erros. Deve-se ter muito cuidado para os valores que passam pelas interfaces serem do tipo correto e estarem dentro dos limites exigidos pelo programa. Os valores podem ser conferidos dentro da rotina que os envia ou na rotina que os recebe, evitando-se, assim, muitos erros.

A checagem, nesse caso, é feita tanto para os valores fornecidos por um usuário como para os lidos de um arquivo. É sempre necessário conferir os valores de entrada numa rotina. As sub-rotinas são projetadas para gerar um conjunto bem definido de saídas, mas não se espera o mesmo dos seres humanos, que podem responder a uma pergunta de diferentes maneiras. Assim, as rotinas que aceitam dados do usuário precisam ser checadas com rigor.

Aconselha-se igual procedimento para todas as rotinas de manipulação de arquivos, pois ocorrem problemas até no momento da leitura dos dados.

Não é comum os programas pararem de rodar em razão dos erros. Isso só acontece quando se desrespeita alguma regra da linguagem (usando ilegalmente um operador, como em RESULTADO = PRIMEIRO$ + SEGUNDO$) ou do sistema operacional (abrindo arquivos demais ao mesmo tempo). Por exemplo, as instruções a seguir parecem constituir um programa correto:

```
10 FOR CONT=1 TO 10
20 SOMA=SOMA+1
30 PRINT CONT, SOMA
40 GOTO 10
50 NEXT CONT
```

No entanto, é um algoritmo sem fim que vai provocar problemas pelo próprio funcionamento da linguagem. Nesse caso, a linguagem (BASIC) possui um controle para acompanhar os loops FOR-NEXT, somando um a cada vez que um novo loop começa.

Neste programa, a linha 50 (com o comando NEXT que diminuiria o contador) nunca é alcançada; assim, o contador aumenta gradualmente até a emissão da mensagem de "estouro" ("overflow"), quando o programa é interrompido pelo interpretador. Erros como este são detectados com facilidade, mas, se aparecerem em rotinas pouco usadas, precisa-se realizar um teste completo para descobri-los.

Um tipo de erro mais sutil é o que permite rodar o programa, mas invalida os resultados. Ocorre, por exemplo, no desenho de formas preenchidas com cor. As rotinas de preenchimento verificam os limites da forma. Quando um dos

Telhado aberto

A ilustração mostra um desenho gerado por computador, em três estágios, sendo preenchido por cor. Um erro deliberado na rotina do desenho deixa um buraco na forma a ser preenchida. Quando o programa chega ao contorno da chaminé e não encontra um limite, o cursor continua pintando fora do contorno. Dependendo do algoritmo de preenchimento usado, isso resulta no preenchimento completo da tela.

limites é alcançado, o computador desvia o cursor e continua desenhando até atingir outro limite. Para uma rotina de preenchimento funcionar, os limites precisam ser bem definidos e completos. Isso significa que não pode haver espaço aberto no contorno da forma, senão a rotina espalha a cor além dos limites.

As versões da linguagem BASIC usadas pela maioria dos micros possibilitam manipular os erros com facilidade, produzindo mensagens claras e concisas. Permitem também dar prosseguimento aos programas com problemas após a alteração dos valores das variáveis no teclado — um recurso útil quando os erros de um programa estão sendo eliminados. Na maioria das versões do BASIC usa-se o comando ON ERROR GOTO com o objetivo de transferir o fluxo de controle para uma rotina especial, que dá um tratamento específico a erros. Para tanto, inclui-se no programa uma linha como:

```
30 ON ERROR GOTO 20000: REM rotinas de
   tratamento de erros
```

Assim, a ocorrência de um erro faz o programa se comportar como se tivesse encontrado a instrução GOTO 20000. Às vezes, a instrução ON ERROR cria duas variáveis: a primeira armazena um número que indica o tipo de erro ocorrido e a outra identifica o número da linha em que o erro foi encontrado. Os nomes dados a essas variáveis e os números dos erros resultantes diferem conforme o equipamento, o que exige consulta ao manual. Uma vez detectado o erro, o fluxo do programa desvia-se para a linha 20000; o número armazenado na variável permite a identificação do erro e, assim, a ação correta é executada.

Um programa bem escrito não tem mais do que uma rotina ON ERROR. Tais rotinas não são capazes de lidar com erros de sintaxe, escassez de memória ou overflow. O máximo que esse recurso oferece é um encerramento organizado do programa, garantindo que todos os arquivos sejam fechados e que o usuário saiba exatamente o que aconteceu.

Alguns erros, como a divisão por zero, podem ser eliminados por essa rotina, mas convém utilizar outro método. Há várias razões para isso:

- A instrução ON ERROR GOTO e a subseqüente volta ao programa principal constituem entrada e saída extras de uma rotina. Isso viola o princípio de programação estruturada, de acordo com o qual uma rotina deve ter apenas um ponto de entrada e um de saída.
- O lugar apropriado para se proteger de uma divisão por zero é a própria rotina que faz a divisão. Não constitui boa prática projetar algoritmos que interrompam o sistema. Se a checagem extra de erro diminuir a velocidade do programa a um grau inaceitável, a rotina deverá ser reprojetada de forma a otimizá-la.
- As rotinas de tratamento de erros tornam-se complicadas se existem cadeias de IF-THEN-ELSE com saídas múltiplas. Elas são restritas pela numeração de linhas do resto do programa e, assim, torna-se necessário reescrevê-las quando se refaz qualquer rotina que se sirva delas. Essas rotinas são difíceis de projetar, testar e corrigir; se contiverem erros, os problemas introduzidos não serão detectados com facilidade, devido ao desvio do fluxo de controle por caminhos imprevisíveis.

OUSADOS CONSERVADORES

O fornecimento regular de componentes permitiu à Commodore invejáveis cifras de vendagem de computadores no mercado internacional. Esse êxito se deve, em grande parte, ao talento de Jack Tramiel e de Chuck Peddle.

A Commodore é uma enorme companhia e, devido a seu volume de produção, pode fazer bons negócios com fornecedores externos: em alguns casos, paga apenas a metade do que seus competidores pagam por chips.

A forte posição da empresa deve muito ao sucesso do CBM PET. Chuck Peddle, responsável por seu design, baseou a máquina no processador 6502, equipou-a com o BASIC da Microsoft e lhe forneceu um editor de tela completo, muito mais fácil de usar do que os painéis simples que os entusiastas vinham utilizando desde o advento do microprocessador, na metade da década de 70. Até hoje, o IBM PC, supostamente da próxima geração, não dispõe, no modelo padrão, de um editor de tela completo.

A Commodore estava em posição privilegiada para construir tal máquina — ela já possuía a MOS Technology, que tinha os direitos de produção do microprocessador 6502. Isso foi fundamental: ambos os competidores do PET — o Apple II e o Tandy TRS-80 — usavam a CPU 6502. Assim, a Commodore podia acompanhar a produção dessas duas companhias. Contudo, o nascimento do PET foi problemático. O presidente da empresa, Jack Tramiel, insistia em que os componentes da memória do PET também fossem da MOS Technology, ao contrário do que queria Peddle. Isso levou a uma disputa interna que resultou na demissão do designer.

Originalmente acondicionado num gabinete de aço, o Commodore PET recebeu depois novo revestimento plástico, que lhe deu uma aparência modernizada. Seu modelo com capacidade de 22 Mbytes armazenados em disco constitui a série 8000.

Apesar da idade, o PET vendia muito bem ainda em 1985. Clientes mais conservadores sentem-se à vontade com ele e não vêem por que mudar seu software para a nova geração de computadores baseados na CPU 8088. Seus fornecedores alegam até mesmo que — para seu próprio espanto — o PET ainda pode competir com o IBM PC e o Apricot.

Por conservadorismo e por um desejo de conter os custos, a empresa nunca se preocupou em ampliar as especificações de sua máquina para principiantes. A maior parte do software original do PET ainda pode rodar nas máquinas de hoje. O BASIC da Microsoft versão 2.0 ainda é, em grande parte, o mesmo da época em que Peddle o adotou.

Isso implica desvantagem para os amadores mais adiantados. O Microsoft 2.0 foi desenvolvido numa época em que os recursos gráficos e de som eram considerados supérfluos em computadores de baixo custo.

Os computadores para amadores que a Commodore lançou recentemente são bem equipados com essas características, mas o BASIC não tem os comandos necessários e exige lento e laborioso uso de comandos POKE para endereçar posições específicas da memória. Isso, no entanto, não é problema com o software em cartucho já pronto, do qual a Commodore oferece grande variedade.

O sobrevivente de Auschwitz

Devido sobretudo ao fornecimento regular de semicondutores da matriz para as fábricas, a Commodore conseguiu superar as dificuldades que forçaram seus competidores a derivar para o mercado de jogos. A Texas e a Mattel retiraram-se do mercado dos micros não profissionais e a Atari não vem se destacando no setor. A Commodore, com o fornecimento garantido de peças a baixo custo, podia forçar a redução dos preços do computador e do software em cartucho. Assim, eliminava a competição e ainda obtinha lucro. Seus cartuchos chegaram a custar um terço do que custavam os de seus competidores. Uma fábrica moderna e automatizada e a disponibilidade de peças a baixo preço foram o resultado de dois decênios de experiência em fabricação.

A Commodore não alcançou essa posição privilegiada por acaso. Ela tem enfrentado situações difíceis e esteve pelo menos duas vezes à beira da falência.

O polonês Jack Tramiel era adolescente quando chegou aos Estados Unidos, após a Segunda Guerra Mundial, como sobrevivente do campo de concentração de Auschwitz. Em 1955, conseguiu formar a CBM (Commodore Business Machines), cujo nome foi escolhido devido à semelhança com o da IBM. A matriz ficava em Toronto, Canadá, onde a companhia iniciou suas atividades como montadora de máquinas de escrever, sob licença da Tchecoslováquia.

Em 1975, após duas décadas de comercialização na área de produtos para escritório, a Commodore fez grandes progressos.

Os construtores
A força motriz da Commodore tem sido seu presidente, Jack Tramiel.

Chuck Peddle
É o homem por trás do projeto do PET e do chip 6502 nele incorporado. Peddle saiu da Commodore para formar sua própria companhia e produzir a máquina para uso profissional Sirius.

Na época, houve uma guerra pelo mercado das calculadoras, vencida pelos japoneses. Mas Tramiel — respeitado e temido por seus métodos de competição — conseguiu atravessar o período. Ele percebeu o potencial de mercado do microcomputador e, em 1976, levou Chuck Peddle para a companhia. Em menos de dez anos, o capital da Commodore aumentou cinqüenta vezes.

A quinta geração

Tramiel fez dela uma formidável empresa de fabricação e comercialização, mas não obteve grande progresso no desenvolvimento de novos produtos.

A filosofia da companhia é: "Nós vendemos para o povo, não para a elite". Tramiel acredita que o cliente comprará o produto que tiver mais a oferecer em troca de seu investimento. Mas as exigências de produção barata e em massa podem agir contra a incorporação de tecnologia mais avançada.

No final de 1982, como boa parte da pequena equipe de pesquisa e desenvolvimento havia deixado a Commodore, ela passou a apoiar-se nas pesquisas realizadas por outras companhias. Iniciou entendimentos com empresas no Extremo Oriente, visando à fabricação de unidades de disco. E tem feito contato com empresas como a Sony, para a aquisição de tecnologias caras da quinta geração: reconhecimento de voz, robôs para uso doméstico e sofisticados dispositivos para armazenamento.

A Commodore chegou mesmo a abordar o criativo projetista Paul Johnson, da Oric, tentando conseguir que ele projetasse um chip para sua nova série de micros.

Na metade da década de 80, a Commodore parecia mais confiante do que nunca e continuava a apostar no preço baixo e na simplicidade. Ela apresentou dois micros (o 264 e o V364) baseados nos novos processadores 7501. O V364 possui um sintetizador de fala com vocabulário de 250 palavras incorporado. Seguindo as tendências atuais, os softwares para processamento de texto, cálculo de folhas eletrônicas e gráficos estarão disponíveis como opcionais.

Os campeões da Commodore

1977
Commodore PET original foi o primeiro microcomputador produzido para o grande público. Após numerosas modificações, ainda vende bem.

1984
O **SX64** é a versão modernizada do 64 em gabinete portátil com tela em cores e unidade de disco.

1982
O **CBM 700** é o substituto das máquinas 8032 e deve elevar os computadores profissionais da Commodore ao nível dos micros mais modernos.

1979
O **Vic-20** foi o primeiro micro de baixo custo da Commodore. Apesar das limitações e da forte competição, ainda é bastante popular.

1981
O **CBM 8032** equipou a série dos PET com capacidade de 80 colunas, dando às máquinas a possibilidade de rodar software profissional.

1983
O **Commodore 64,** com tela de 40 colunas e memória de 64 K, superou as limitações do Vic.

1980
Com o **SuperPET** (ou **CBM 9000**), tentou-se produzir uma versão profissional do PET.

PALETA ELETRÔNICA

O tablete gráfico Koala existe em versões para o Apple e o IBM PC. Aqui examinaremos a versão para o Commodore 64, que utiliza plenamente os recursos gráficos de alta resolução desse microcomputador.

O Commodore 64 não foi suprido com periféricos para gráficos de alta qualidade, mas a companhia americana Koala Technology desenvolveu um tablete gráfico que permite aos usuários desse micro fácil acesso a sua capacidade de criar gráficos em alta resolução.

Medindo apenas 20,5 x 16 cm, o tablete gráfico Koala é leve e compacto. No centro, há um quadrado de fibra de carbono, de 11 x 11 cm, que cobre uma membrana sensível ao toque. Esta, pressionada com um dedo ou um lápis, guia o cursor pela tela.

Duas camadas de fios condutores — uma no eixo horizontal, outra no vertical — constituem a membrana. Quando ela é pressionada, o tablete gráfico detecta os fios que estão em contato e envia as coordenadas resultantes para o computador. Acima da membrana há dois botões, dos quais um deve ser pressionado quando o usuário deseja colorir um ponto da tela ou selecionar uma das várias opções de cores. Qualquer dos botões pode ser usado — provavelmente para facilitar o manuseio por canhotos e destros.

Colorindo

O tablete gráfico Koala conecta-se ao computador pela saída para joystick e o Koala Painter (o software necessário para o tablete funcionar) é carregado a partir de um disquete. Todas as opções disponíveis aparecem então na tela.

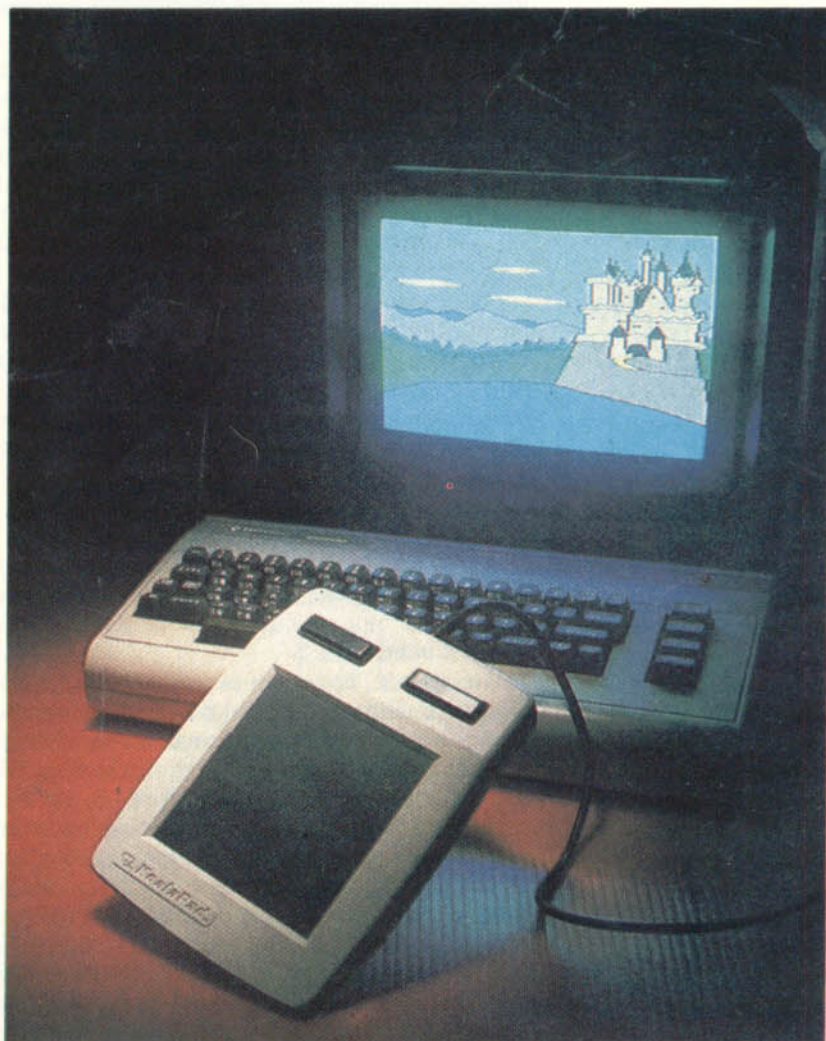
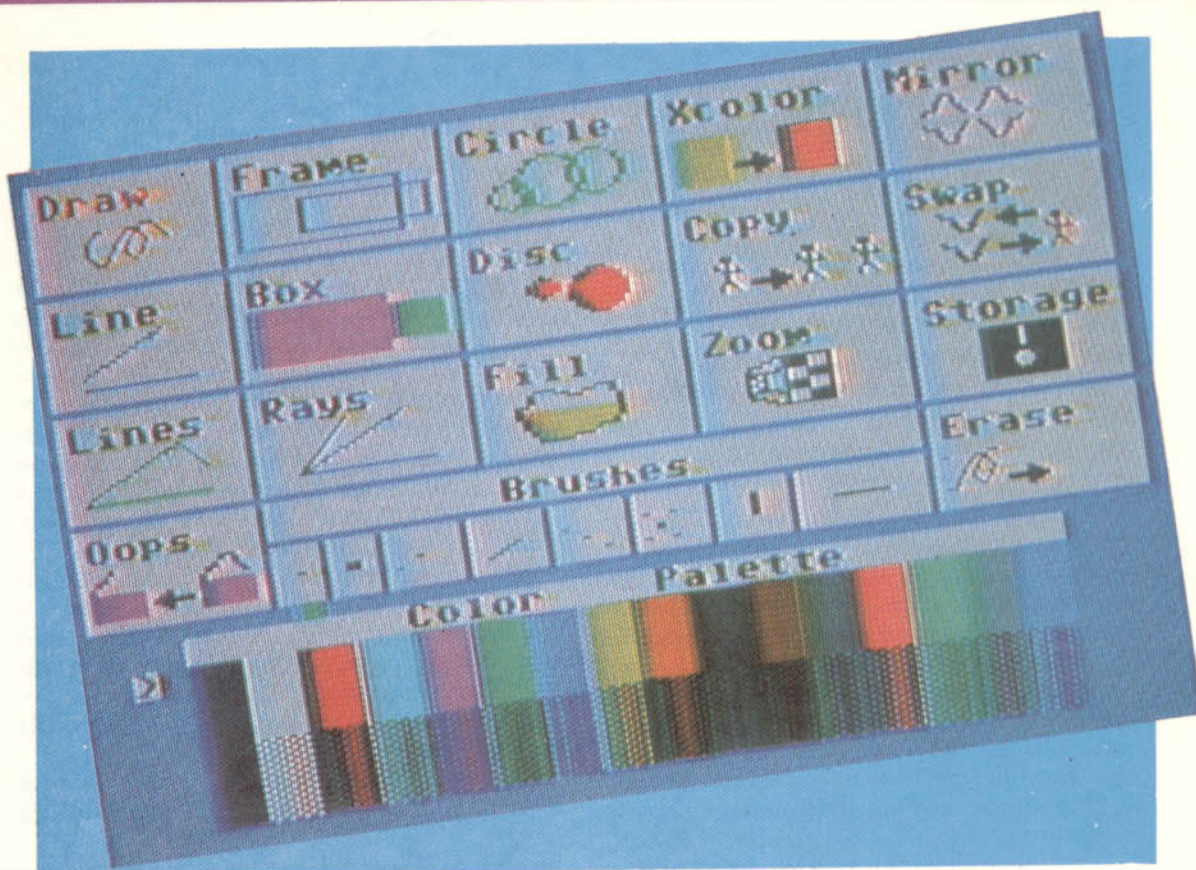
Na parte inferior fica a "paleta", contendo dezesseis cores "puras" e outras tantas "mistas". As misturas são obtidas colorindo-se pixels alternados com diferentes matizes, o que produz esse efeito.

Acima da paleta há oito quadros contendo os "pincéis" (BRUSHES): formatos registráveis na tela, que variam de um único pixel até combinações de pixels e linhas. Ao redor dos pincéis há várias opções para o desenho de linhas ou formas na tela. O usuário as seleciona pressionando a membrana do tablete gráfico, direcionando um cursor em forma de seta. Quando a seta aponta para a opção desejada, a pressão num dos botões de seleção faz com que ela entre em ação. Uma luz intermitente brilha no retângulo indicativo da opção, para lembrar ao usuário qual o modo que está sendo usado.

O tablete gráfico Koala oferece recursos para produzir linhas isoladas, raios (linhas traçadas a partir de um único ponto), quadros e círculos. Blocos de cor podem ser acrescentados usando-se a opção BOX (quadrados coloridos) ou a opção DISC (círculos coloridos). Consegue-se manipular melhor o colorido usando o comando FILL para preencher uma área delimitada com a cor escolhida. As cores podem ser alteradas pelo uso do comando X-COLOUR.

Figuras idênticas são desenhadas simultaneamente usando MIRROR. Esse comando divide a tela em quatro seções, com o cursor restrito ao quadro superior esquerdo. Qualquer coisa traçada naquele quadro será automaticamente copiada nas áreas correspondentes dos outros três quartos.

Conseguem-se traçar desenhos detalhados a nível de pixel utilizando o comando ZOOM. O

Koala-pad
Telas de gráficos complexos e sofisticados são construídas em poucas horas quando se usa este software dirigido por menus de fácil compreensão.

Escolhas artísticas
O menu principal do tablete gráfico Koala consiste em quadros contendo o nome e o símbolo visual da opção. Apesar de os símbolos pretenderem auxiliar a compreensão, algumas das ilustrações são confusas. Para selecionar uma opção, move-se o cursor para um quadro e pressiona-se o botão de seleção. O nome do quadro selecionado ilumina-se intermitentemente para lembrar ao usuário que modo está sendo acionado.

usuário pode escolher qualquer trecho do desenho, que na parte inferior aparece como uma "janela" ampliada. Mostram-se quadrados do tamanho de um caractere de 8 x 8 pixels. Corrigidas, essas figuras são então colocadas em qualquer lugar da tela, por meio do comando COPY; isso permite definir na tela uma área cujo conteúdo é copiado em qualquer outra posição.

Selecionadas essas opções, o cursor é movido para fora da tela e o usuário pressiona o botão de seleção. A tela torna-se, então, o suporte para o quadro a ser pintado. Movendo o cursor e pressionando o botão de seleção, pode-se desenhar linhas e formas em qualquer parte da tela normal para gráficos do Commodore 64.

Prós e contras

A qualidade dos gráficos obtidos com esse dispositivo é excelente, alcançando o mesmo nível das telas de alta resolução produzidas por softwares comerciais.

No entanto, como em outros pacotes de gráficos, a qualidade do comando DRAW (para desenhos a mão livre) decepciona. A resolução da matriz da membrana não se equipara à da tela de alta resolução do Commodore 64 e, assim, o lápis (ou a ponta do dedo) do usuário raramente está sobre uma interseção da matriz. Na maioria das vezes atinge dois pontos ao mesmo tempo e o computador, ao interpretar isso, assinala um ponto que não se encontra na posição desejada.

Como resultado, não raro o que deveria ser uma linha reta acaba semelhando um rabisco desorganizado. Outra crítica é a de que, com exceção do comando ZOOM, não há opção para mudar de cor sem que seja necessário retornar ao menu principal. Mas essas são falhas secundárias, compensadas pela velocidade com que se executam os comandos LINE e FILL.

Outra limitação do software reside no método usado para eliminar erros, que emprega o comando OOPS, acessado a partir do menu principal. O uso do comando OOPS apaga todo o trabalho realizado pelo usuário desde a última vez que ele saiu do menu principal. Isso significa que meia hora de trabalho pode ser apagada devido a um único erro. Uma alternativa que se coloca é apagar um erro usando o comando ZOOM e corrigindo-o pixel por pixel, o que, no caso de um DISC ou BOX mal colocado, pode demorar algum tempo. Uma modificação bem-vinda seria restringir a extensão do comando OOPS à ultima pressão do botão de seleção e não mais à última saída do menu principal.

É possível usar o tablete gráfico Koala como um joystick, e assim se consegue acessá-lo a partir de seus próprios programas. A posição do cursor é obtida em BASIC pelo comando PEEK, usando as localizações 54297 e 54298 para as coordenadas x e y.

Após sua gravação em disco, as telas são facilmente transferidas para os programas do próprio usuário, o que permite o desenvolvimento de jogos com texto na parte inferior da tela e desenhos na superior.

Com o tablete gráfico Koala são gravadas e acessadas até dezesseis diferentes telas de 8 Kbytes num disquete. Como não é possível car-

regar uma tela diretamente a partir do disquete para a memória de tela, a Koala Technology fornece no guia do usuário um programa que possibilita transferir telas da área onde estão armazenadas para a memória de tela.

A resolução máxima de uma tela de Koala Painter gravada é de 255 x 255 pixels, se estiver incluída num programa em BASIC. Só se alcançam graus de resolução mais altos com o uso de linguagem de máquina. Essa limitação pode causar problemas, já que a tela de alta resolução do Commodore 64 tem o tamanho de 320 x 200 pixels.

O tablete gráfico Koala se restringe a um máximo de 255 pixels porque esse é o maior número que pode ser endereçado por 1 byte — o endereçamento de uma tela completa exigiria endereços de 16 bits. Uma das prováveis conseqüências disso é a perda de parte da imagem, pois, quando não está sendo exposta, a tela fica armazenada em um lugar determinado da RAM disponível para o usuário.

Portanto, a tela de 8 Kbytes, juntamente com as informações sobre as cores, deve ser transferida de sua localização na RAM do usuário para a memória de tela de alta resolução, sempre a partir da localização 55296.

O tablete gráfico Koala constitui investimento compensador para quem possui uma unidade de disco e gosta de arte ou de jogos de aventura.

Construindo um quadro

Esta série de etapas demonstra algumas das características do tablete gráfico Koala. Primeiramente, usando o comando ZOOM, o artista desenha cada letra da palavra "pleasure". Então, usando o comando COPY, ele as dispõe na parte inferior da tela, em seqüência. Depois, por meio do comando LINE, traça uma linha que atravessa o topo do letreiro e desenha o contorno da estrela para produzir efeito tridimensional. Blocos de cores são então acrescentados com o uso do comando FILL. Outras características puderam ser incluídas recorrendo aos comandos DISC e CIRCLE.

PRIMEIRAS PALAVRAS

Principais recursos do PASCAL
- Livre formatação do texto do programa.
- Regras flexíveis para nomear os objetos.
- Permissão de definir novas "palavras-chaves".
- Sintaxe simples e coerente.
- Definição dos dados.
- Estrutura de programação em módulos.
- Controle flexível dos dados e dos processos.
- Recorrência natural.
- Excelente diagnose de erro na compilação.
- Um compilador pequeno e muito eficiente.

Blaise Pascal, 1623-1662
Matemático, cientista e filósofo, Blaise Pascal inventou a calculadora mecânica em 1642. O nome da linguagem PASCAL foi uma homenagem a sua contribuição para a ciência da computação.

O PASCAL é a mais influente linguagem de programação de alto nível. Embora algumas pessoas considerem sua sintaxe muito rígida, ele foi especialmente desenvolvido para o ensino da boa técnica de programação.

A linguagem PASCAL foi projetada pelo pesquisador suíço Niklaus Wirth, por volta de 1970. Seu nome é uma homenagem ao matemático e filósofo francês do século XVII, Blaise Pascal, o inventor da primeira calculadora com quatro funções. O PASCAL deriva diretamente do ALGOL 60 e constituiu uma resposta de Wirth ao ALGOL 68, uma linguagem por demais complexa e extensa. Ele pretendia fazer com que o PASCAL:

- permitisse a expressão exata dos conceitos e estruturas da programação;
- demonstrasse que uma linguagem concisa e de alto nível, dispondo de um conjunto flexível de tipos de dados, de instruções e de recursos para a estruturação de programas, pode servir como instrumento de uso geral para a solução de problemas;
- facilitasse a elaboração de métodos para organizar programas extensos e para manipular complexos projetos de software de modo seguro e garantido;
- tivesse amplos recursos para verificação de erros, especialmente durante a compilação, diminuindo assim ao máximo os erros de programação e fornecendo um veículo excelente para o ensino da programação; e
- pudesse ser usado eficientemente em micros.

Todos os objetivos do projeto foram plenamente alcançados — em geral, um pequeno compilador PASCAL ocupa apenas 24 Kbytes e é duas vezes mais eficiente que o FORTRAN (famoso por sua velocidade). Embora o PASCAL tenha um vocabulário restrito e seja fácil de aprender (possui apenas 35 "palavras-chaves" ou "reservadas", ao contrário do BASIC, com mais de uma centena, na maioria de suas versões), nem por isso é menos poderoso. O PASCAL é uma linguagem muito mais expressiva, tanto pelo modo de escrita dos algoritmos como pela facilidade para se descrever os dados de forma simples e coerente, seja qual for a complexidade destes.

Talvez a sua maior vantagem, no entanto, seja a de oferecer um método simples e coerente para a expressão de algoritmos extremamente poderosos. Afinal, o modo como consideramos os problemas de computação é o fator que mais facilita a sua resolução. A liberdade natural de expressão torna o PASCAL um excelente instrumento para a solução de problemas. Além disso, a linguagem apresenta vários outros recursos: por ser uma linguagem compilada, não apenas os programas são executados com muito mais rapidez, como também a memória não é desnecessariamente ocupada pelo programa-fonte e um interpretador de linguagem — mas apenas pelo código-objeto compilado.

Toda a filosofia da linguagem se resume em proteger o usuário de sua própria desatenção e evitar que rode um programa cheio de erros. Isso pode parecer estranho, especialmente para os programadores em linguagem BASIC. Muitas vezes, contudo, quanto mais rápido se elabora um programa em BASIC, mais tempo se leva para fazê-lo funcionar.

No caso de programas extensos é realmente muito mais fácil programar em PASCAL do que em BASIC. Embora muitas vezes o PASCAL seja criticado por suas regras estritas, são elas que o tornam tão eficaz na detecção e identificação de erros de programação. Afinal, essa disciplina imposta não passa de uma exigência necessária para a elaboração de qualquer programa com eficácia e segurança. Em conseqüência, economiza-se um tempo enorme que seria gasto na depuração de programas que "quase funcionam".

Para o programador habituado ao BASIC, a diferença mais notável dos programas em PASCAL, de qualquer extensão, é a grande quantidade de definições e declarações que parecem desprovidas de qualquer sentido. O primeiro 1,5 m de um programa extenso em PASCAL parece, de fato, não ter qualquer utilidade. Isso em parte resulta de que, embora você possa acrescentar à linguagem suas próprias palavras, todo o procedimento inicial é necessário para que o PASCAL entenda o que elas significam. Assim, enquanto no BASIC primeiro se define o programa principal (usando-se instruções do tipo GOSUB 5000) e, depois, as sub-rotinas, o PASCAL possibilita definir novos comandos como, por exemplo, LIMPARTELA ou PAUSA (tantos segundos) no início do programa e usá-los no procedimento principal. Por exemplo:

```
BEGIN
  LIMPARTELA;
  WRITE ('ALO!');
  PAUSA (3);
  ...ETC.
```

O PASCAL incentiva uma abordagem sistemática, mas muito prática, da programação de com-

putadores. Ele proporciona ao usuário uma visão conceitual de altíssimo nível a respeito de um sistema de computação, permitindo que dados e processos sejam definidos e expressos de modo natural e lógico. Isso garante um alto grau de portabilidade e segurança intrínsecos, e tanto a detecção de erros como seu diagnóstico são excelentes. De modo geral, o PASCAL teve mais influência sobre outras linguagens de computação — e no projeto de software — do que qualquer outra linguagem de programação.

O diagrama abaixo, que descreve a genealogia das linguagens de alto nível, mostra apenas as influências mais importantes das linguagens compiladas que empregam comandos imperativos; assim, não inclui as linguagens LISP, PROLOG, nem qualquer outra linguagem funcional. O tronco principal de influências tem início com o ALGOL 60 e será difícil encontrar uma única linguagem moderna digna de nota que não derive, direta ou indiretamente, do PASCAL. Conseqüentemente, um conhecimento sólido do PASCAL será muito útil para quem quiser entender as linguagens da década de 90 — como MODULA-2, OCCAM ou ADA.

Apesar de ter sido desenvolvido como um instrumento para o ensino de programação, o PASCAL foi adotado em muitos aplicativos e sistemas comerciais, e tem sido usado no projeto dos mais variados tipos de software, como pacotes financeiros e compiladores de linguagem. O sistema operacional ''p-system'', famoso por sua portabilidade e projetado no final da década de 70 pela UCSD (Universidade da Califórnia em San Diego), foi desenvolvido e escrito em PASCAL. O software para os micros Lisa e Macintosh, da Apple, inclusive seus sistemas operacionais, também foi elaborado principalmente em PASCAL ou em sua derivada, o CLASCAL. Em meados da década seguinte, a equipe da UCSD começou a projetar os compiladores para o MODULA-2 e outros sistemas, além de aplicativos para vários fabricantes de software — todos em PASCAL ou em linguagens derivadas.

O mais surpreendente é que tudo isso aconteceu sem qualquer investimento por parte de al-

Genealogia das linguagens imperativas de alto nível 1954-1982

4
1955
6
7
8
9
1960
1
2
3
4
1965
6
7
8
9
1970
1
2
3
4
1975
6
7
8
9
1980
1
2

FORTRAN

1954-57 FORTRAN (FORmula TRANslator) Linguagem científica para o processamento numérico e de matrizes.

COBOL

1959-60 COBOL (COmmon Business Oriented Language) Linguagem comercial útil na manipulação de arquivos.

ALGOL 60

1958-60 ALGOL 60 (ALGOrithmic Language) Influente linguagem para expressão de algoritmos.

PL/1

1963-64 PL/1 (Programming Language 1) Tentativa de juntar as três linguagens.

ALGOL 68

1963-68 ALGOL 68 Versão gigante e poderosa do ALGOL, definida por uma comissão européia de cientistas.

PASCAL

1968-71 PASCAL Linguagem concisa e bem estruturada, projetada para ser poderosa e eficaz, apesar de simples.

BCPL

1969 BCPL (Basic Combined Programming Language) Linguagem que alia a estrutura do ALGOL ao poder de acesso à máquina propiciado pelo ASSEMBLER.

C

1974-78 C Linguagem para sistemas usada no projeto do sistema operacional UNIX.

PASCAL CONCORRENTE

1975 PASCAL CONCORRENTE Versão ampliada para multiprogramação.

PASCAL UCSD

1977 PASCAL UCSD Versão ampliada para sistemas.

MODULA-2

1977-80 MODULA-2 Nova linguagem modular de Wirth que vem conquistando a preferência da comunidade informática.

ADA

1978-82 ADA Linguagem adotada pelo Pentágono para todos os contratos governamentais dos Estados Unidos.

OCCAM

1981 OCCAM Linguagem concisa e eficiente para multiprogramação.

PASCAL ISO

1980-82 PASCAL ISO Versão do PASCAL padronizada internacionalmente.

guma empresa importante. O sucesso do PASCAL deveu-se apenas a seus próprios méritos e não por ser fonte de lucros para os grandes fabricantes.

Enfrentando o compilador

O primeiro problema que se encontra no uso de uma linguagem compilada está em aprender a lidar com o processo em vários estágios, mesmo que o programa seja pequeno. Em primeiro lugar, o programa-fonte deve ser introduzido mediante algum tipo de editor ou processador de texto. Gravado o texto-fonte em fita ou disco, o compilador da linguagem é carregado e recebe instruções para compilar o programa-fonte em código de máquina. Por fim, esse "arquivo-objeto" deve ser relocalizado, e suas partes — que foram compiladas separadamente — devem ser interligadas ("linkeditadas") às rotinas que serão necessárias para o processamento. Talvez o programa possa ser carregado e executado sem necessidade de trabalho complementar, mas se o compilador usar um pseudocódigo (código intermediário) será preciso valer-se de um interpretador na execução do programa.

Se esse processo parece muito complicado, console-se com o fato de que todas as versões de PASCAL para micros evitam em grande parte esses problemas. Assim, na execução do programa, o texto-fonte, o compilador e o programa-objeto podem estar simultaneamente presentes na memória. A eficiência e o pequeno tamanho do PASCAL tornam isso possível e o uso desse sistema não é, em geral, mais complexo que o de um BASIC residente. Somente no caso de programas mais extensos será preciso recorrer aos processos mais complexos acima descritos.

Cada sistema dispõe de seu próprio conjunto de comandos para controlar o editor e o compilador, e você deve consultar a documentação apropriada para utilizar esses recursos. Muitas vezes bastará um simples E para editar, C para compilar e R para rodar. O que nos interessa é a sintaxe correta a ser empregada em cada programa, seja ele simples ou complexo. Felizmente, a linguagem PASCAL é tão bem padronizada que haverá pouca ou nenhuma necessidade de adaptar os programas para cada versão, como ocorre no BASIC (embora existam algumas exceções de menor importância). Assim, examine esse primeiro programa completo em PASCAL:

```
PROGRAM PRIMEIRO (OUTPUT);
CONST
  MENSAGEM = 'PROGRAMACAO PASCAL';
BEGIN
  WRITE (MENSAGEM)
END.
```

Experimente carregar, compilar e executar esse programa antes de analisá-lo em detalhe. Se receber alguma mensagem de erro do compilador, leia-a com cuidado e veja se pode detectar o erro. Cada símbolo deve ser digitado exatamente como mostramos. Dê atenção especial ao ponto-e-vírgula no final da primeira e da terceira linhas e, também, ao ponto-final no fim do programa. Executando corretamente esse programa, você obterá a seguinte mensagem na tela:

Compilação popular
Uma implementação profissional completa do PASCAL pode muitas vezes custar mais que um micro, mas em meados da década de 80 foi lançada uma série de compiladores de preço relativamente acessível.

```
PROGRAMACAO PASCAL
```

Esse programa pode ser banal, mas serve como exemplo da forma geral que todos os módulos PASCAL (programa, procedimento ou função) deverão apresentar. Ela divide-se em três partes distintas:

1. O cabeçalho; neste caso, o nome do programa.
2. Declarações e definições dos dados; no exemplo, uma única definição de constante.
3. O "núcleo", que contém todas as instruções a ser executadas.

Análise sintática

As regras sintáticas do PASCAL, pelo menos quanto aos elementos básicos da linguagem, são melhor definidas por meio de "diagramas de sintaxe". Estes são como o mapa rodoviário de um sistema em sentido único. Uma rotina legítima no diagrama vai da parte superior esquerda até a parte inferior direita e todo bloco por que passarmos é: ou uma "entidade sintática" (isto é, que representa a si mesma) contida em um bloco de cantos arredondados, ou algum item descrito em outro ponto por meio de um diagrama de sintaxe separado (indicado por um bloco retangular).

Diagrama de sintaxe BEGIN-INSTRUÇÃO

Com relação ao diagrama geral de um programa, note que as palavras BEGIN e END fazem parte do vocabulário reservado do PASCAL e não exigem diagramas complementares para definir seu significado. No PASCAL existem apenas 35 palavras reservadas com significado predeterminado. O programa acima utiliza somente quatro delas — PROGRAM, CONST (abreviação de constante), BEGIN e END. A palavra que segue PROGRAM é um "identificador", que indica o nome escolhido por você para o programa.

Se começarmos com uma letra e usarmos somente letras ou dígitos, teremos uma quantidade quase infinita de nomes aceitáveis. Entretanto, como era de se esperar, não é permitido usar uma palavra reservada. Por exemplo:

```
NOME
PASCAL
PROGRAMAUM
N
XYZ123
ENDERECO12
JOSESILVA
IDENTIFICADORBEMLONGO
```

são todas aceitáveis, enquanto:

```
PROG-1
DEZ%
AND
123QUATRO
SE PARECE
```

não são legítimas, ou porque contêm símbolos que não são alfanuméricos, ou porque começam por um número, ou ainda (no caso de AND) porque são uma das palavras reservadas. O último exemplo não é válido por causa do espaço entre as palavras, embora cada uma delas (SE e PARECE) seja aceitável. Em PASCAL, como na linguagem comum, não há diferença de significado entre letras minúsculas e maiúsculas, embora algumas versões não padronizadas exijam que as palavras reservadas estejam em maiúscula.

Além dos espaços e do final de linha, há um outro elemento da sintaxe do PASCAL que pode ser usado como um separador — o comentário. Este pode aparecer em qualquer ponto do texto, menos, é claro, no meio de palavras. Os comentários são indicados por chaves { }.

Sejamos um pouco mais ousados e examinemos algo mais parecido com um verdadeiro programa em PASCAL:

```
PROGRAM PROGRAMADOIS (INPUT, OUTPUT);
  {FORNECE O QUADRADO DE UM NUMERO}
CONST
  QUESTAO = 'ENTRE COM UM NUMERO:';
VAR
  NUMERO : INTEGER;
BEGIN
  WRITELN;
  WRITELN;
  WRITE (QUESTAO);
  READ (NUMERO);
  WRITELN (NUMERO, ' AO QUADRADO DO ',
    NUMERO * NUMERO)
END.
```

Observe que agora incluímos o identificador INPUT no título. O PASCAL exige os identificadores INPUT e OUTPUT, para a indicação dos arquivos externos com os quais o programa se comunicará. Geralmente, nos micros, eles serão o teclado e o vídeo, respectivamente. Com a inclusão do INPUT, podemos agora ler por meio do teclado, utilizando o procedimento padrão do PASCAL, READ. Como acontece com WRITE, qualquer "parâmetro" deve ser apresentado entre parênteses.

A memória utilizada para armazenar esses parâmetros é reservada por meio da declaração VAR — no exemplo, para um único número inteiro. Ao contrário do BASIC, que distingue somente entre números e strings (por meio do cifrão — $ — após o identificador), o PASCAL dispõe de um conjunto quase ilimitado de tipos de dados. Por isso, como o compilador precisa saber quanto de memória deve reservar para cada item dos dados, você sempre precisa declarar cada uma das variáveis em uma declaração VAR.

O cursor permanecerá posicionado imediatamente após o indicador do sistema, como se tivéssemos usado uma instrução PRINT do BASIC, com um ponto-e-vírgula no fim. Isso não é um erro, mas exatamente aquilo que solicitamos ao empregarmos o procedimento WRITE incorporado no PASCAL. Sempre que uma nova linha for necessária, usamos o procedimento alternativo WRITELN (LN é uma contração da palavra LINE). Uma instrução WRITELN simples, sem parâmetros, irá simplesmente criar uma nova linha.

O PASCAL permite diferenciar entre dados que variam e itens que permanecem constantes durante a execução do programa. Assim, observe que a definição CONST usa um sinal de igualdade, enquanto a declaração VAR usa dois pontos.

As 35 palavras reservadas do PASCAL

							OR	REPEAT	UNTIL
							PACKED	SET	VAR
AND	CASE	DO	END	FUNCTION	IN	NIL	PROCEDURE	THEN	WHILE
ARRAY	CONST	DOWNTO	FILE	GOTO	LABEL	NOT	PROGRAM	TO	WITH
BEGIN	DIV	ELSE	FOR	IF	MOD	OF	RECORD	TYPE	

MOUSE E ICONOGRAFIA

O editor de ícones permite a criação de novos ícones ou a adaptação dos existentes.

Menu principal: escolhe-se com o cursor um dos quatro programas aplicativos.

Para criar uma figura, primeiramente se desenha o contorno.

Depois escolhem-se no menu os diversos tipos de linhas disponíveis.

Paralelamente à tecnologia do mouse, desenvolveu-se a fabricação do software que o acompanha. O AMX, por exemplo, inclui, além do mouse, o programa AMX Art, projetado para o BBC Micro.

O pacote AMX, da firma inglesa Advanced Memory Systems, criado para o BBC Micro, inclui o próprio mouse, acompanhado de um cabo para conectá-lo ao micro, um chip de ROM, dois manuais e software em cassete ou em disco.

Embora fornecido em cassete e em disco, o AMX é dirigido especialmente aos proprietários de micros com discos.

Fabricado no Japão, o AMX é um mouse comum, produzido em material plástico preto, com uma esfera de metal na base, para detecção de movimento. O mouse cabe na palma da mão, permitindo que três dedos se apóiem nos botões seletores de ação: Move, Execute e Cancel.

A decisão de incorporar uma esfera metálica — e não de borracha dura — mostrou-se contraproducente, pois esse material tende a escorregar em superfícies como madeira envernizada ou revestida em fórmica. Assim, é aconselhável colocar um pedaço de papel sobre a superfície lisa, para que a esfera role sem problemas.

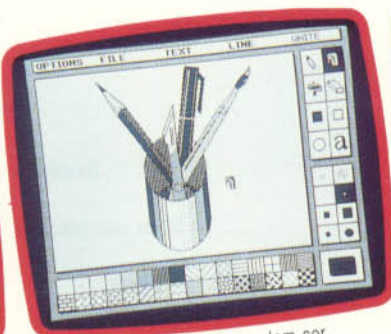
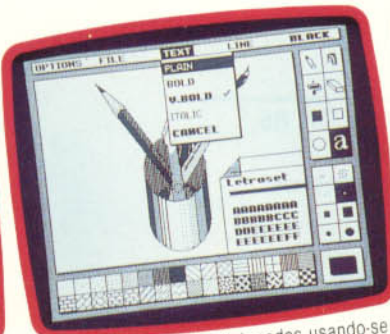
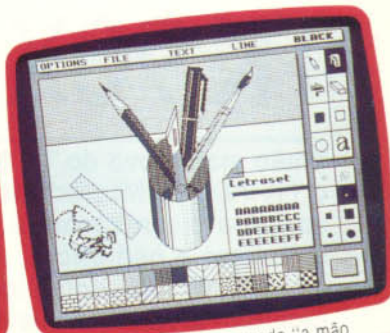
Dentro do mouse, dois cilindros ladeiam a esfera. Na extremidade de cada cilindro há um disco com ranhuras que passa por uma célula fotoelétrica. O disco intercepta um feixe luminoso, criando impulsos ópticos captados pela célula e transmitidos como comandos direcionais ao computador. Desse modo, o computador sabe sempre a localização, o tipo de movimento e a velocidade do mouse.

O software residente no chip da ROM situado na lateral faz a comunicação com o computador. Além das rotinas "primitivas" para uso dos pacotes de software, a ROM fornece várias rotinas que possibilitam a utilização do mouse em programas feitos pelo usuário. Essas rotinas incluem uma que fornece a posição do mouse, em coordenadas de tela; outra que lê os botões seletores; e uma terceira que anima os ícones (figuras) na tela. Usando essas rotinas — todas selecionadas pelo sistema operacional do BBC Micro —, o usuário pode escrever, com rapidez e facilidade, programas em BASIC que aproveitam ao máximo o controle do mouse.

Outras rotinas possibilitam o uso do mouse com pacotes comerciais de software. O movimento do mouse substitui as teclas do cursor e os três botões seletores podem ser configurados para agir como qualquer das três teclas, inclusive [Shift] e [CTRL]. Dessa forma, é possível usar o mouse para eliminar, mover e copiar partes de texto num processador.

No entanto, o valor do mouse só fica evidente quando se usa o software aplicativo desenvolvido para ele. O pacote AMX inclui dois progra-

Mouse AMX
O mouse AMX vem como parte de um pacote completo, que inclui o próprio mouse, um chip de ROM contendo o software de controle, vários softwares aplicativos em cassete ou disco, um manual do usuário e outro que ilustra aplicações artísticas.

Os contornos são preenchidos com uma série de motivos sombreados.

Texturas complementares podem ser acrescentadas.

Letras e palavras são adicionadas usando-se o modo texto e escolhendo o tipo desejado.

Retoques finais exigem o modo "a mão livre".

mas desse tipo, que demonstram a facilidade e a simplicidade do uso do mouse. O primeiro desses é um programa em BASIC para desenhar ícones incorporáveis em qualquer programa feito pelo usuário.

O programa de desenho Icon apresenta ao usuário um tabuleiro de dezesseis por dezesseis quadrados, nos quais ele poderá desenhar seu ícone preenchendo os quadrados desejados — tudo por meio do mouse.

Com esse programa, as únicas ocasiões para usar necessariamente o teclado ocorrem quando você for digitar o nome do arquivo em que gravou o grupo de ícones ou quando digitar o nome de arquivo de um conjunto já existente, para carregá-lo na memória. Devido ao alcance do controle do mouse, ele é fácil de usar e o desenho de ícones complexos e de efeito realista se faz com rapidez e simplicidade.

A melhor demonstração do potencial do mouse é dada pelo AMX Art, também incluído no pacote AMX. O AMX Art é um programa para desenho de gráficos com todos os recursos controlados pelo mouse. Equipara-se ao pacote Macpaint do Macintosh da Apple quanto à facilidade de utilização.

O uso do programa

Executa-se o desenho na região central da tela — o programa usa uma tela em duas cores, de 320 por 256 pixels. Do lado direito estão dois menus de ícones. O menu superior é empregado para escolher entre várias operações: desenhar linhas, apagar, esfumaçar e preencher áreas.

Cada uma dessas funções aparece na tela representada por uma imagem gráfica facilmente identificável: um lápis, uma borracha, um rolo de pintura. Determinada ação é escolhida movendo-se o cursor, por meio do mouse, sobre a imagem desejada e pressionando-se em seguida um dos botões seletores, o que dá ao cursor a forma do ícone escolhido.

A figura pode ser elaborada deslocando-se o cursor para a área a desenhar. Usando os botões seletores, você executa qualquer das funções do desenho. Como o movimento do cursor reproduz com precisão o movimento do mouse e o uso do programa é simples, o excelente manual que acompanha o AMX Art se torna quase desnecessário.

Outro menu que aparece na lateral da tela seleciona, pelo mesmo procedimento, as espessuras de linha e as formas. Na parte inferior da tela está um terceiro menu para a escolha de padrões sombreados que preenchem as formas. Eles variam do preto a complexos motivos entrelaçados. Como a tela só tem duas cores — devido a limitações de memória —, esses padrões são fornecidos para distinguir as áreas com diferentes tipos de sombreado.

Outras opções — como gravar ou carregar os desenhos criados, ativar os comandos do sistema ou listar um dump da memória — podem ser feitas por meio de menus que aparecem na parte superior da tela. Só as opções principais desses menus são apresentadas na tela; quando você posiciona o cursor sobre o item escolhido e pressiona um botão seletor, o restante do menu se desenrola verticalmente sobre a tela, podendo cobrir parte do desenho. Depois que você fez suas opções com o mouse, o menu desaparece, deixando a imagem como antes.

O usuário pode obter composições complexas, precisas e inovadoras já ao utilizar o programa pela primeira vez. Pouco tempo depois sente-se tão à vontade com esse pacote como se estivesse usando lápis e papel — e o AMX Art não quebra a ponta, não borra e pode ser reproduzido quantas vezes se deseje.

A AMX pretende ampliar o pacote mouse, transformando-o num sistema completo para o BBC Micro. Um gerenciador financeiro, semelhante no funcionamento ao que é usado nos computadores Macintosh e Lisa, estava sendo desenvolvido em meados da década de 80. A empresa também planejava um banco de dados controlado por mouse e outros aperfeiçoamentos para o programa AMX Art, incorporando cores e o efeito de zoom. O mouse e os menus pull-down serão com certeza incluídos em muitos programas aplicativos populares.

O mouse AMX e o programa AMX Art, assim como outros programas baseados no mouse e nos ícones, valorizam o BBC Micro para o usuário que estiver pensando em expandir os recursos de sua máquina.

RATOS, JANELAS E ÍCONES

Mouse: Dispositivo de operação manual conectado eletronicamente ao computador e dotado de esfera giratória na parte inferior. Movido sobre uma superfície plana, ocasiona um deslocamento correspondente do cursor na tela. O nome mouse (camundongo) deriva de sua aparência: uma caixinha, em geral preta, com um longo fio que se assemelha à cauda de um rato.
Menu pull-down: Menu que aparece na parte superior da tela, mostrando apenas as opções principais. Quando se seleciona um item, surgem as alternativas referentes a ele numa janela que se abre verticalmente sobre a tela. Terminadas as escolhas, o menu volta a ocupar apenas a linha superior da tela.
Ícones: Símbolos gráficos apresentados num programa para indicar as opções de um menu de modo visual, em vez de usar palavras ou combinações de teclas.

MEMÓRIA SELECIONADA

Dump: Listagem de todos os conteúdos de uma área da memória.

TK 90X

Irmão mais novo do TK 83 e TK 85, o TK 90X, da Microdigital, combina a vantagem de inúmeros softwares e periféricos disponíveis com comandos que facilitam a tarefa do usuário.

Portátil, capaz de funcionar ligado a aparelhos comuns de televisão, o TK 90X, da Microdigital, oferece imagens de alta resolução gráfica em oito cores básicas e geração de som pela tevê. Compatível em hardware e software com o ZX Spectrum +, da Sinclair, esse equipamento não apresenta nenhuma relação com seus antecessores TK 83 e TK 85. Considerado um avanço da linha Sinclair, o TK 90X incorpora uma interface para joystick — tipo Atari/Onyx Jr. —, com quatro posições de movimentação e uma de disparo, controladas também pelo teclado, e caracteres de acentuação para a língua portuguesa e cedilha. Tais recursos gráficos, no entanto, exigem o acionamento de três teclas.

Esse modelo da Microdigital destaca-se também pela quantidade de periféricos disponíveis; possui saídas para Wafadrive, microdrive, unidade de disco, caneta óptica, gravador de EPROM, expansão de memória de 32 K, sintetizador de 10 oitavas, com 130 semitons, traçador digital e conversor analógico-digital.

A grande quantidade de software disponível, no Brasil e no exterior, é outro ponto digno de nota. Os programas abrangem as áreas educacional, profissional e de lazer. O TK 90X roda também programas nas linguagens LOGO, PASCAL, FORTH e BASIC compilado.

O teclado tipo QWERTY compõe-se de quarenta teclas equivalentes a funções e comandos BASIC, compreendendo 37 funções matemáticas lógicas e científicas, além de 36 caracteres alfanuméricos, 26 caracteres minúsculos/maiúsculos, oito caracteres gráficos e mais oito em vídeo reverso.

O TK 90X apresenta diversos comandos que facilitam a tarefa do usuário. Para efeitos visuais, por exemplo, o comando INK permite escolher a cor do caractere mostrado no vídeo, enquanto o comando BORDER oferece a possibilidade de traçar margens coloridas, independentemente da cor no centro. Outro recurso é o comando FLASH (cintilador), usado para destacar informações na tela.

Utilizando fitas cassete comuns para armazenamento de dados, o TK 90X dispõe ainda de comandos que permitem ao usuário acompanhar um programa em andamento, bem como integrar dois ou mais programas BASIC na memória da máquina. Há comandos específicos para gravar e carregar programas, e também para ler dados e telas.

Saída para expansões

Fonte de alimentação

Dissipador de calor

Controlador de teclado e gravador

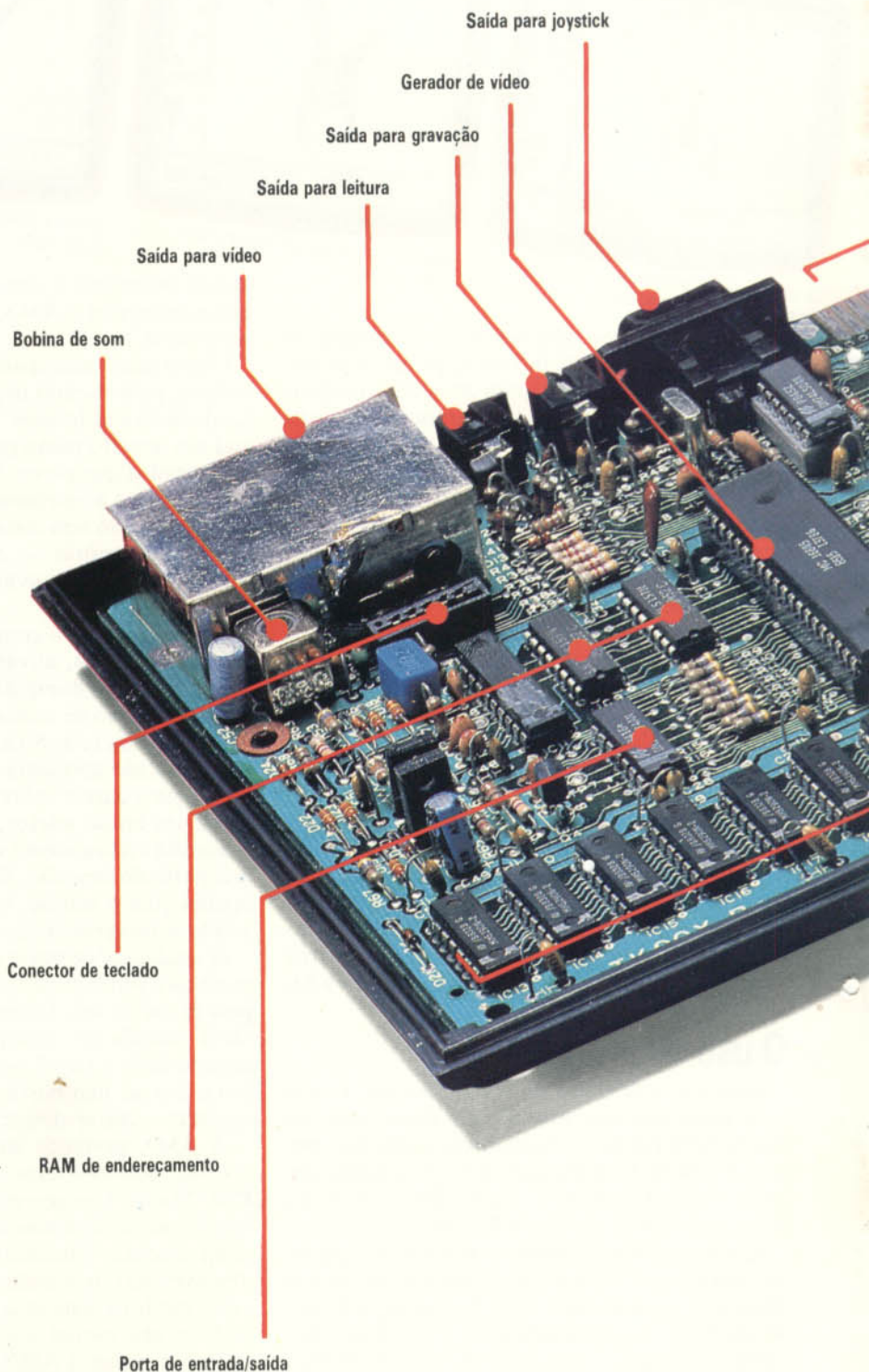
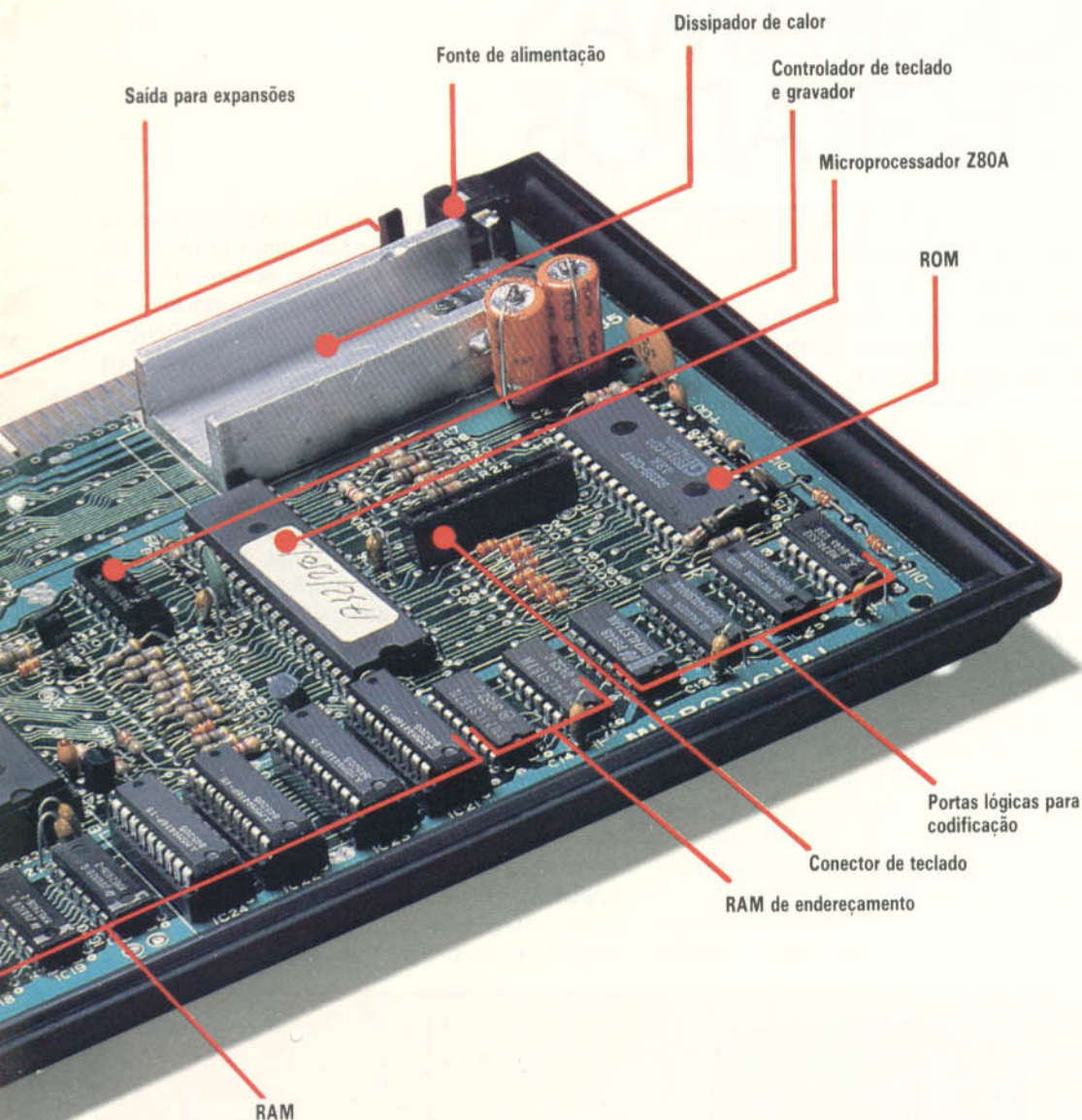
Microprocessador Z80A

ROM

Portas lógicas para codificação

Conector de teclado

RAM de endereçamento

RAM

TK 90X

MICROPROCESSADOR

Z80A, de 8 bits.

CLOCK

3,58 kHz.

MEMÓRIA

16 K de ROM e 16 ou 48 K de RAM.

VÍDEO

Saída para tevê em cores, sistema PAL-M ou P&B. Modo texto: 24 linhas de 32 caracteres; modo gráfico: 256 x 192 pixels. Trabalha em modo invertido e apresenta recursos de flashing e brightness.

TECLADO

Tipo QWERTY, com quarenta teclas equivalentes a funções e comandos BASIC, 93 funções BASIC pré-programadas, 37 funções matemáticas lógicas e científicas, 36 caracteres alfanuméricos, 26 caracteres alfanuméricos minúsculos/maiúsculos, oito caracteres gráficos, oito caracteres gráficos invertidos, 21 caracteres gráficos criados pelo usuário.

LINGUAGENS

BASIC e ASSEMBLER.

PERIFÉRICOS

Programador de EPROM, expansão de RAM de 32 K, saída serial RS232C para comunicação de dados, saída paralela para impressora e saída para joystick.

DOCUMENTAÇÃO

Manual de operação e programa tutorial, em fita, que ensina a trabalhar com o teclado e a operar o micro.

Software diversificado
Compatível com o ZX Spectrum Plus, da Sinclair, o TK 90X executa grande número de programas — só na Inglaterra existem quatrocentos aplicativos educacionais para o aparelho, alguns dos quais já traduzidos pela Microdigital.

PROGRAMAS INTEGRADOS

Os programas Lotus 1-2-3, Symphony e Xchange são softwares integrados, projetados especialmente para microcomputadores profissionais. Mas suas técnicas se aplicam também a máquinas de uso pessoal.

O software integrado exige que o usuário tenha acesso instantâneo a todas as tarefas de que venha a precisar; e também que os procedimentos operacionais permaneçam inalterados, qualquer que seja a aplicação em uso; e, ainda, que as informações possam ser transferidas facilmente de uma aplicação a outra. Há várias maneiras de preencher esses requisitos.

O programa Lotus 1-2-3, da empresa americana Lotus, integra planilha financeira, banco de dados e geração de gráficos em cores. Operado por menus de fácil utilização, foi um campeão de vendas em todo o mundo, em 1984. Utiliza o formato da planilha financeira, na qual os números e as fórmulas são introduzidos em células definidas por linhas horizontais e colunas verticais, que podem ser modificadas.

O programa oferece vários recursos extras — daí sua utilização para diversos fins. As células da planilha podem ser usadas para armazenar informações não financeiras, como nomes e números de telefone. Uma parte específica da planilha é utilizada como tabela de pormenores relevantes; por exemplo, uma lista de clientes e seus números de conta. Essa área será usada como um pequeno banco de dados, pois o Lotus 1-2-3 oferece funções de pesquisa e reorganização das informações.

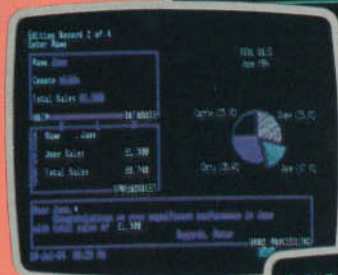
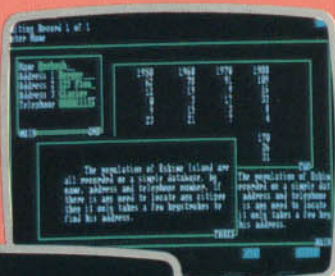
Também é possível tomar uma série de células com dados numéricos e utilizar o programa para mostrar essa informação com diferentes tipos de gráficos em cores: linhas, barras, barras superpostas, ou círculo dividido em setores. Esse recurso elimina a necessidade de um programa gerador de gráficos separado.

O programa Lotus 1-2-3 tem ainda recursos para lidar com textos, o que permite seu emprego na redação de memorandos e cartas, mas a limitação de memória impede seu uso como um verdadeiro processador de texto.

Essa combinação de recursos faz do Lotus 1-2-3 o único programa necessário para grande número de usuários. Toda a informação que as aplicações requerem está contida numa planilha: assim, chega-se a resultados que seriam impossíveis com os programas tradicionais.

Variações sinfônicas
O Symphony integra quatro aplicações, ocupando toda a memória disponível com uma gigantesca planilha. Os dados são mostrados na tela em janelas, que os interpretam de acordo com a função em uso: processamento de texto, planilha financeira, banco de dados ou gráfico. Esse sistema facilita a visualização e resolve o problema do intercâmbio dos dados entre as várias aplicações, mas requer grande quantidade de RAM.

Symphony

Banco de dados

Planilha

Gerador de gráficos

Processador de textos

Vamos supor, por exemplo, que um usuário do Lotus 1-2-3 administre várias bancas de jornais, em diferentes pontos da cidade. Ele precisa registrar as vendas semanais, mensais, trimestrais e anuais, referentes a cada uma das locações. A melhor forma de fazer isso é pôr a localização de cada banca e suas vendas numa planilha. As fórmulas são elaboradas de tal modo que o usuário só precisa mudar os números relativos às receitas semanais de cada banca. Os outros valores cujas fórmulas dependem daqueles números serão reajustados automaticamente.

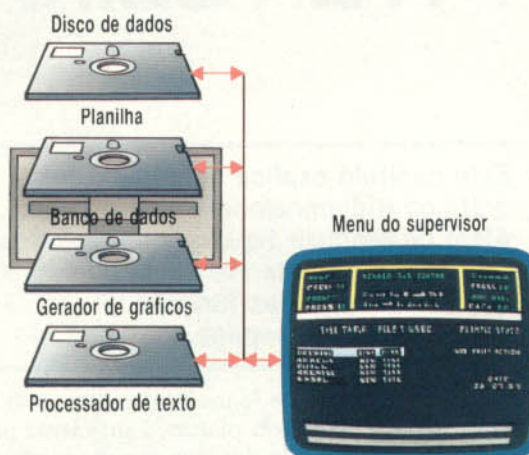
Se o usuário quiser colocar as bancas em ordem de vendas — ficando a que vende mais no alto da lista —, encontrará recursos no Lotus 1-2-3. Inicialmente, introduzem-se os nomes das bancas em ordem alfabética, mas eles precisam ser reclassificados a cada semana, quando novos valores são recebidos. Com o Lotus 1-2-3, isso é rápido e fácil. O dono das bancas pode pedir ao programa um quadro semanal que mostre o desempenho de cada uma. Alguns toques no teclado permitem que essa informação seja acessada da planilha/banco de dados, mostrada na tela em forma de gráfico e impressa.

O programa da Lotus que se seguiu ao 1-2-3 — o Symphony — obedece ao mesmo princípio de basear as aplicações em formato de planilha, mas, além disso, permite a divisão da tela em diversas áreas (as "janelas"), focalizando diferentes partes da planilha. Cada janela ganha uma formatação apropriada, de acordo com a informação que mostra.

Quando a informação é um texto, a janela toma a forma de uma pequena tela de processamento de texto, com margens e tabulações bem demarcadas. Quando se precisa de um gráfico, outra janela mostra os eixos com suas escalas e seus títulos. Cada entrada de informação do banco de dados ganha sua própria tela, que se parece com um registro num fichário.

O Symphony tem quatro aplicações importantes, todas trabalhando ao mesmo tempo, e, assim, podem ser mostradas em separado na tela. Tal como o Lotus 1-2-3, memorizam determinadas seqüências de comandos, para que o usuário automatize as operações mais utilizadas. As seqüências são ativadas por pequenos programas chamados "macrocomandos".

O Symphony também inclui sua própria linguagem de programação de alto nível. Os programas são armazenados na planilha da mesma forma que quaisquer outros dados e acessam todas as operações disponíveis. Desse modo, se você precisar fazer um sistema de faturamento ou controle de estoque, pode programá-lo em linguagem Symphony. Os programas tornam-se parte de suas aplicações padrão.

Uma vez familiarizado com o Symphony, será mais simples programar em sua própria linguagem de comandos do que usar outra, pois o Symphony lida facilmente com tarefas como desenhar gráficos ou pesquisar e mesmo organizar dados.

Livre intercâmbio
No programa Xchange (Intercâmbio), da firma inglesa Psion, a planilha, o processador de texto, o banco de dados e o gerador de gráficos ficam em discos separados. Quando se executa um dos programas, os dados criados são tratados como tarefas pelo supervisor, que controla e mantém na memória até dez tarefas simultâneas. O usuário passa de uma tarefa para outra por meio de um menu. Conforme a opção do menu, o supervisor carrega e executa o aplicativo apropriado. Os dados passam de uma tarefa para outra pelos comandos EXPORT e IMPORT.

O Symphony é apenas um dos muitos softwares integrados disponíveis no mercado e tem no Framework, da empresa americana Ashton Tate, um forte concorrente. Os dois são caros e exigem grande capacidade de memória. O primeiro opera com 320 Kbytes, mas precisa de 512 para o uso de todos os seus recursos. Já o Framework requer um mínimo de 256 Kbytes. Em função disso ambos só rodam em micros de 16 bits, como o IBM PC e compatíveis.

Nem o Symphony nem o Framework têm necessidade de que segmentos do programa fiquem armazenados em discos, caso da maioria dos programas aplicativos comerciais. Os chips de memória dos computadores continuam se tornando cada vez mais baratos; não é absurdo, portanto, que as equipes que desenvolvem software sustentem que os usuários terão grandes extensões de memória à disposição. Na prática, porém, isso ainda não acontece, e pode ser que tais aplicações com uso de memória extensa demorem algum tempo para se fazerem mais comuns. Embora um programa como o Symphony estabeleça novos padrões, ainda está sujeito às limitações do hardware. Em outras palavras, o Symphony só faz o que faz por ser um programa extenso e cuidadosamente elaborado.

A TERCEIRA DIMENSÃO

Este capítulo explica como construir gráficos tridimensionais com o micro. Além de produzir figuras interessantes, tal recurso permite visualizar o comportamento das funções matemáticas no espaço.

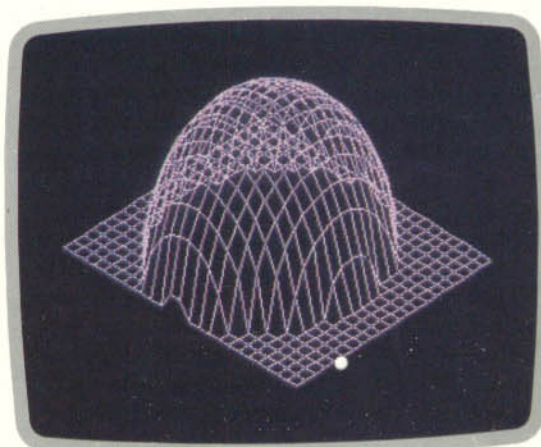
A simples menção de senos e tangentes, sem falar nos gráficos em três planos, é suficiente para balançar a segurança dos que se acham afortunadamente livres da matemática do colégio. No entanto, com a ajuda de um computador, esses tópicos podem tornar-se agradáveis, mesmo sem que se compreendam inteiramente seus princípios.

Graças a sua capacidade gráfica, a maioria dos microcomputadores é ideal para a apresentação de gráficos formados a partir de equações matemáticas. Escritas com símbolos matemáticos, tais equações produzem desenhos atraentes, capazes de interessar mesmo os que odeiam matemática.

Os desenhos aqui apresentados foram produzidos num micro doméstico, usando os programas listados, todos calculados com gráficos em três dimensões. Não se trata, na verdade, de gráficos tridimensionais, uma vez que os computadores só apresentam imagens em duas dimensões. No caso, "tridimensional" refere-se à ilusão de profundidade obtida com a superposição de diversos gráficos bidimensionais.

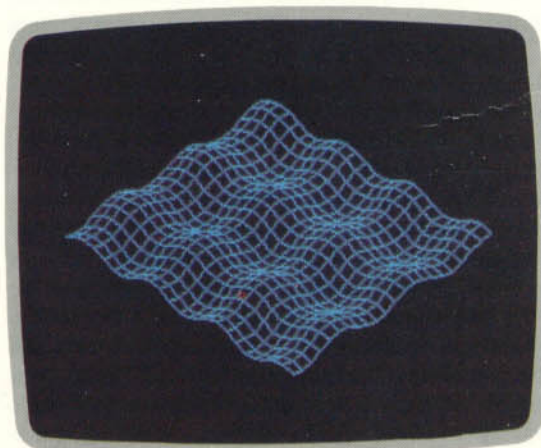
Os programas listados neste capítulo calculam os valores de uma equação com duas variáveis, X e Z. O resultado, Y, é calculado para cada valor de X e Z. Os valores de Y são usados para desenhar na tela os pontos situados no eixo vertical. Os pontos bem próximos ligam-se com pequenas linhas retas, que produzem, no conjunto, a impressão de uma linha curva. Um plano representa as coordenadas X e Y, mantendo-se Z constante, ao passo que o plano que o intersecciona perpendicularmente representa Y e Z, com X constante. A construção dos gráficos ajuda a entender funções matemáticas complicadas.

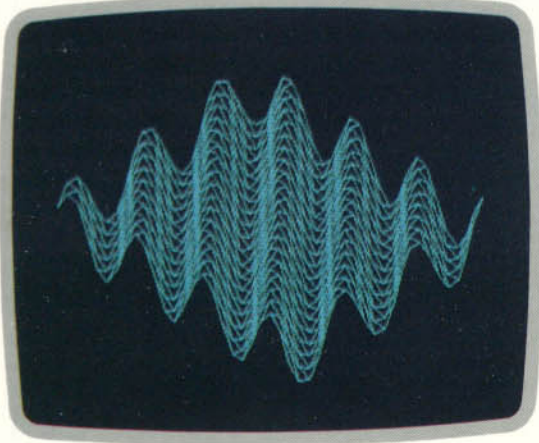
Esses gráficos também podem servir de estímulo a pessoas que não se interessam muito pela matemática. É divertido (e bem difícil) tentar montar uma equação que resulte numa determinada forma.

Para alterar o gráfico apresentado, é necessário mudar a função na linha 390 do programa em BASIC. Algumas funções são mais complicadas, exigindo, por isso, mais de uma linha de programa; se for esse o caso, pode-se usar todas as linhas com numeração entre 321 e 399.

```
380  C=50-X*X-Z*Z
390  Y=SQR(C*(SGN(C)+1))/30
```

Além de escolher uma função que resulte numa figura interessante, você deve cuidar para que os valores produzidos não sejam tão grandes que o gráfico ultrapasse os limites da tela. Para mantê-lo dentro dos limites, a função talvez tenha de ser dividida por um número grande.

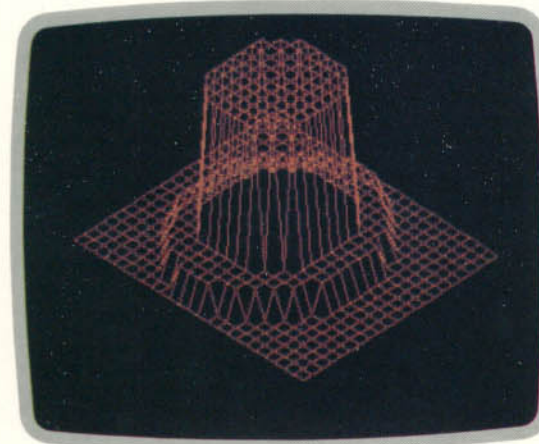
As versões deste programa podem ser usadas por microcomputadores de várias linhas. Para ajudar na conversão, o programa foi feito de modo que a primeira parte prepare uma diagramação padronizada da tela. Assim, uma equação que funciona em determinada máquina também deve funcionar em outras. A segunda parte do programa é usada para calcular e armazenar os valores que determinam os pontos do gráfico, mantendo-os numa matriz. Os cálculos dependem da função escolhida e podem demorar vários minutos; enquanto isso, o computador daria a impressão de que não está fazendo nada. Poupa-se tempo calculando a função antes. Se

Criação de formas
Insira ou modifique no programa básico as linhas mostradas embaixo de cada ilustração, a fim de produzir estes gráficos em três dimensões.

```
390  Y=(SIN(X)+COS(Z))/60
```

os cálculos fossem efetuados enquanto as linhas eram traçadas, o programa tomaria quase o dobro do tempo para traçar o gráfico.

390 Y=SIN(X+Z)/24

Na listagem encontram-se diversas funções diferentes para você experimentar. As ilustrações mostram os resultados esperáveis. Você também pode tentar desenvolver seus próprios gráficos inserindo diferentes funções no programa. Tome cuidado ao fazer isso; é preciso ter certeza de que o gráfico caberá na tela e de que não será tentada nenhuma operação matemática ilegal. Os dois erros mais comuns são tentar dividir por zero (que dá infinito) e tentar extrair a raiz quadrada de um número negativo (que não é um número real).

Para evitar a divisão por zero, acrescente uma constante muito pequena (por exemplo, 0,00001) a qualquer variável que possa tornar-se zero. A única forma de proteger-se contra raízes quadradas de números negativos é usar a função ABS (valor absoluto) para tornar todos os números positivos antes de extrair a raiz quadrada.

Alguns gráficos interessantes resultam de funções matemáticas comuns, tais como SIN (seno), COS (co-seno) e LOG (logaritmo). Outros podem ser conseguidos usando funções só encontradas em computadores — experimente INT (inteiro), SGN (módulo) e ABS.

380 C=X*X+Z*Z+0.00001
390 Y=SGN (INT(13/C))/3+ SGN(INT(35/C))/15

Este programa aceita aperfeiçoamentos diversos. Você pode tentar adaptá-lo de maneira que qualquer função seja colocada em escala automaticamente, a fim de enquadrar-se nos limites da tela. Ou, também, marcar os pontos numa terceira direção, dando curvas para X e Z, enquanto Y é mantido constante (o que é mais complicado). Mas, mesmo usando o programa como está escrito, você verá que é muito divertido trabalhar com as mais simples equações que se pode imaginar. Os resultados certamente irão surpreendê-lo.

Figuras e planos

```
100 REM ********************************
110 REM **                            **
120 REM **   GRAFICO TRIDIMENSIONAL   **
130 REM **                            **
140 REM **       PARA O APPLE II      **
150 REM **                            **
160 REM ********************************
165 REM
170 LARGURA=280:ALTURA=190:VA=-1
180 XFAL=5:ZFAL=3
190 TAMANHO=INT(LARGURA/XFAL/2)
200 PROF=INT(ALTURA/ZFAL/3)
210 HOME:HTAB10:VTAB3:PRINT"CALCULANDO"
220 REM
240 REM ********************************
245 REM **      CALCULA GRAFICO       **
250 REM **    OBSERVAR TEXTO PARA     **
255 REM **         VARIACOES          **
260 REM ********************************
265 REM
270 INICIO=5
280 DIM G(TAMANHO,PROF)
290 FOR A= -PROF/2 TO PROF/2
300 FOR B= -TAMANHO/2 TO TAMANHO/2
310 VTAB23:HTAB03:PRINT"A>";A:
    VTAB23:HTAB12:PRINT"B>";B
320 X=A*18/TAMANHO:Z=B*18/PROF
390 Y=(SIN(X)+COS(Z))/60
400 G(B+TAMANHO/2,A+PROF/2)=Y*VA*ALTURA
410 NEXT B:NEXT A
420 REM
440 REM ********************************
450 REM **     DESENHA PLANO X-Y      **
460 REM ********************************
465 REM
470 HGR2:HCOLOR=7
480 FOR Z=1 TO PROF
490 XBASE=XFAL*Z
500 ZBASE=ALTURA/2+Z*ZFAL+INICIO*VA
510 XVELHO=XBASE+XFAL
520 ZVELHO=191-ZBASE+ZFAL+G(1,Z)
530 FOR X=1 TO TAMANHO
540 XNOVO=XBASE+X*XFAL
550 ZNOVO=191-ZBASE+X*ZFAL+G(X,Z)
560 HPLOT XVELHO,ZVELHO TO XNOVO,ZNOVO
570 XVELHO=XNOVO:ZVELHO=ZNOVO
580 NEXT X:NEXT Z
590 REM
600 REM ********************************
610 REM **     DESENHA PLANO Z-Y      **
620 REM ********************************
630 REM
640 FOR X=1 TO TAMANHO
650 XBASE=XFAL*X+PROF*XFAL
660 ZBASE=ALTURA/2-X*ZFAL+PROF*ZFAL+
    INICIO*VA
670 ZVELHO=191-ZBASE+ZFAL+G(X,PROF-1)
680 XVELHO=XBASE-XFAL
690 FORZ=0 TO PROF-1
700 XNOVO=XBASE-Z*XFAL
710 ZNOVO=191-ZBASE+Z*ZFAL+G(X,PROF-Z)
720 HPLOTXVELHO,ZVELHO TO XNOVO,ZNOVO
730 XVELHO=XNOVO:ZVELHO=ZNOVO
740 NEXT Z:NEXT X
750 GET FIM$
760 TEXT:HOME:END
```

TESTE DE BANCADA

Ligar um aparelho para ver se funciona implica o risco de contato acidental de dois circuitos condutores. Evite danificar delicados componentes montando um aparelho de teste e um multímetro.

A primeira etapa na verificação de qualquer montagem de componentes num circuito testa a eficiência das soldas nas ligações. Às vezes, uma solda feita a temperatura muito baixa parece aceitável, mas não está fazendo nenhuma ligação interna. Um ligeiro puxão talvez revele isso. Se você a executou corretamente, a ligação não se partirá em suas mãos; mas, se isso acontecer, será melhor que ocorra nessa etapa.

Pode-se encontrar o tipo mais simples de aparelho de teste de contato em lojas de ferramentas e de acessórios para automóveis. Vendido com o nome de "teste de ignição", ou outro similar, é utilizado para determinar se o contato no distribuidor está aberto ou fechado.

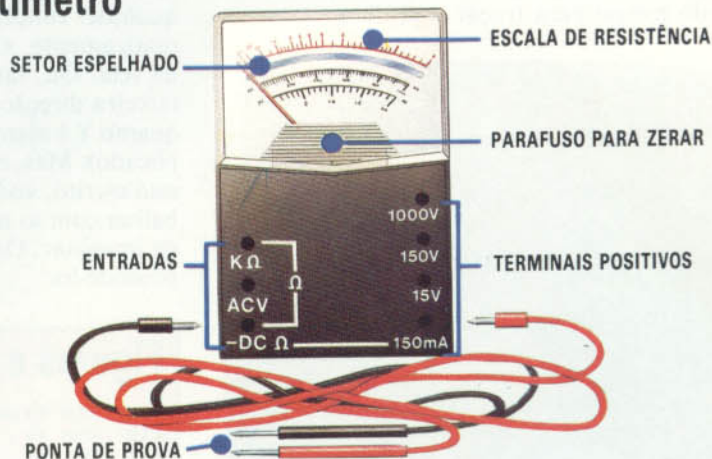
Para manutenção e pequenos reparos em micros, a garra jacaré — que em geral se encontra num dos terminais do aparelho — não é muito útil. Elimina-se esse inconveniente soldando um pedaço pontiagudo de ferro na garra, junto com fita isolante, e usando-o como teste.

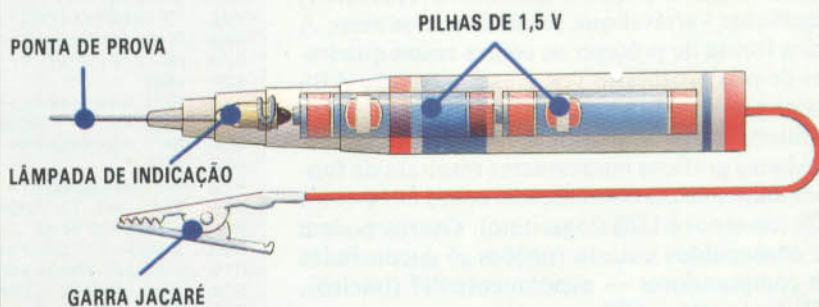
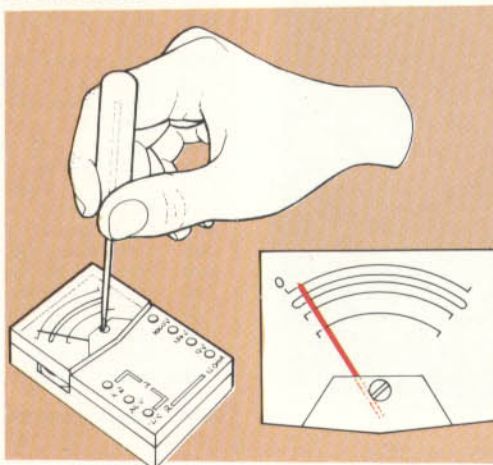
Você monta um aparelho semelhante com duas pilhas de 1,5 V, uma pequena lâmpada de 3 V e alguma extensão de fio. Alternativa melhor, no entanto, é um pequeno multímetro. Funcionando também como ohmímetro, ele pode testar não só a condutividade como a resistência. A unidade de medida da resistência é o ohm (símbolo Ω), nome do físico alemão George Ohm, que descobriu o fenômeno no século XIX.

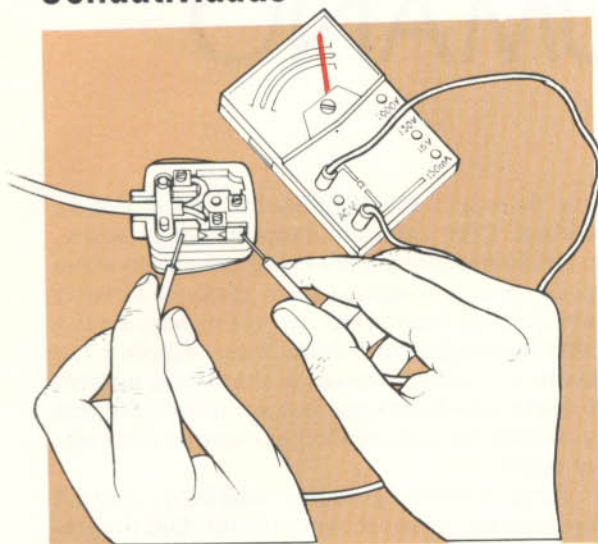
A resistência é função da área da seção transversal do fio que conduz a corrente, mas também pode ser introduzida artificialmente, por meio de resistores.

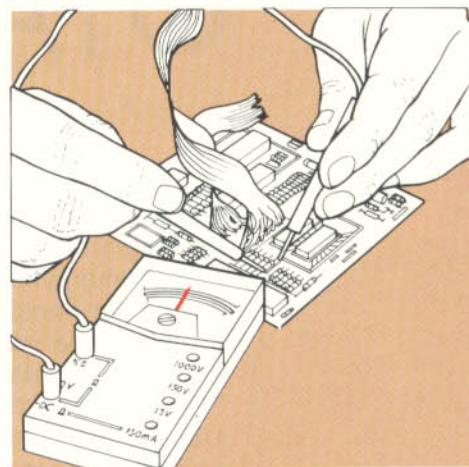
Para nossa tarefa, uma ligação adequada oferece resistência mínima à passagem da pequena corrente que aciona o medidor ou aparelho de teste de condutividade. Assim, a luz deve brilhar com intensidade máxima ou a agulha do mostrador defletir totalmente. Qualquer reação em menor escala indica ligação malfeita.

Além de medir condutividade, o multímetro tem duas funções: indica a corrente (em ampères) e a tensão (em volts). A diferença de potencial (tensão) entre dois pontos de um circuito que conduz uma corrente de 1 A e que dissipa 1 W de energia é de 1 V

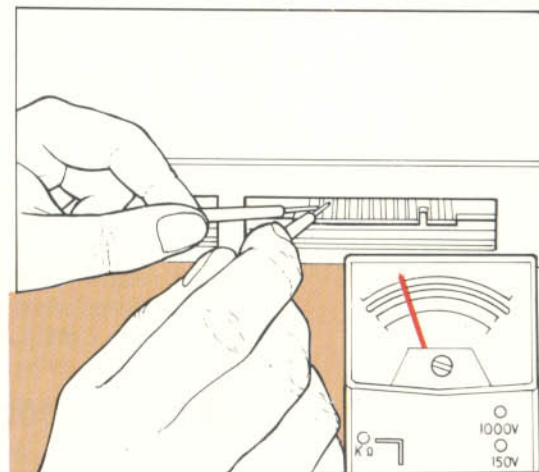
Multímetro

Variam muito os preços dos aparelhos que testam a condutividade e a resistência, medidas em ohms; a intensidade da corrente, medida em ampères; e a diferença de potencial, medida em volts. Mas só existem dois modos de apresentar seus resultados: o analógico e o digital. Instrumentos com bobina móvel que exibem seus dados por meio de uma agulha que se move (defletora) numa escala, de modo análogo ao aumento ou à diminuição do valor medido, são bem mais baratos que suas versões digitais.

Circuito de teste

Zerando o medidor

Variações e imprecisões
Medidores analógicos, nos quais a agulha de indicação se move pelo mostrador, devem permitir calibração. Em geral, isso se faz por meio de um parafuso colocado na base da agulha. O setor espelhado atrás da agulha dá ao observador a certeza de uma leitura acurada. Também o circuito que mede a resistência deve ser ajustável, para levar em conta outras variações e imprecisões.

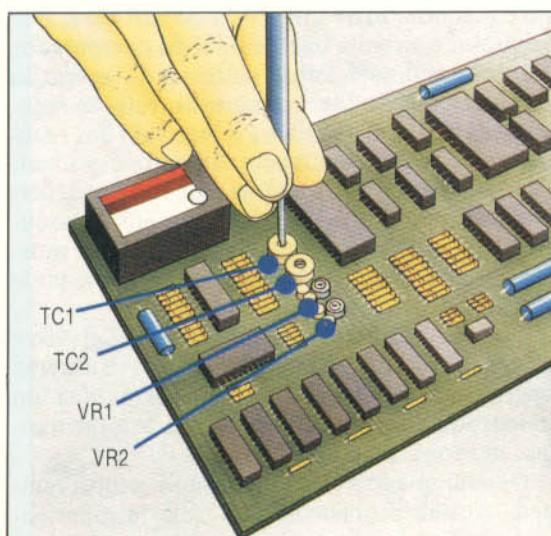
Condutividade

Resistência

Condutividade e resistência
Um dos exemplos mais comuns de falta de condutividade ocorre nos fusíveis. O fusível é projetado para se queimar e abrir o circuito, quando se sobrecarrega. Na ausência de teste de condutividade, a solução usual é a troca do fusível velho por um novo. O teste de condutividade indica também onde está o problema e o multímetro mede o valor da resistência.

Conector de lâmina
A figura mostra o meio físico pelo qual os periféricos se conectam ao computador. Algumas máquinas, como o CP 200, têm somente essa entrada. Outras, como o CP 500, têm muitas. Conectores de lâmina são apenas um tipo de ligação de entrada e saída.

Voltagem
Um bom exemplo do uso do multímetro como instrumento de teste de voltagem: localizar a fonte de energia do conector de lâmina de seu computador. Consulte o manual e localize os pinos de 0, +5 e +12 V. Encoste o terminal negativo do medidor no pino de 0 V. A leitura resultante lhe dirá quão precisa é a regulagem do circuito de controle dentro do computador e seu suprimento de energia.

Garra jacaré
Esse instrumento impreciso converte-se numa boa ponta de prova quando se solda um pedaço pontiagudo de ferro entre as garras.

Ajuste do Spectrum

Os proprietários dos primeiros Spectrum podem melhorar a qualidade da imagem da tevê. Dois resistores variáveis VR1 e VR2 controlam a proporção vermelho-verde e azul-amarelo, respectivamente.

Os capacitores TC1 e TC2 controlam o brilho dos caracteres e a intensidade das cores. Cinco parafusos mantêm unido o gabinete do Spectrum. São fáceis de localizar, na parte inferior do aparelho. Removendo-os, você expõe a PCI (Placa de Circuito Impresso). O procedimento para o ajuste dos aparelhos da linha Sinclair é bem semelhante, mas lembre-se do aviso ao lado, sobre a garantia de seu microcomputador.

CUIDADO!

A garantia de seu microcomputador será invalidada se qualquer pessoa que não o fabricante ou seu agente autorizado abrir o gabinete.

O FATOR HUMANO

Um aspecto importante no projeto de programas é a interação homem/máquina. Os fatores que devem ser considerados ao se projetar essa interação são aqui investigados.

Durante muitos anos, a programação de computadores foi um tópico misterioso, entendido apenas por profissionais dedicados ao assunto. Antes do advento do microcomputador com teclado semelhante ao da máquina de escrever, dava-se entrada aos programas um byte por vez, acionando interruptores no painel frontal do computador ou perfurando fitas de papel.

O usuário de hoje é, ao contrário, uma criatura mimada. Os fabricantes não ficam esperando que ele esbarre na linguagem de máquina, tanto que a expressão "amigável ao usuário" *(user-friendly)* foi criada para indicar que os micros podem ser usados e programados por pessoas sem experiência. Em 1982, o Comitê Alvey, em seu relatório "Programa para tecnologia avançada da informação", identificou a ligação entre homem e máquina como uma das quatro principais áreas de pesquisa e desenvolvimento.

Influências sobre o diálogo

Além das restrições impostas pelos dispositivos de entrada e saída usados, o diálogo entre usuário e máquina sofre influência do software. Por exemplo, o sistema operacional do computador controla muitos detalhes da tela e da operação do teclado. A rapidez com que as teclas se repetem quando pressionadas e o intervalo das repetições são determinados pelo sistema operacional. Esse sistema também acumula toques, a fim de permitir ao computador estocar caracteres introduzidos mais rapidamente do que possam ser mostrados. Isso afeta a velocidade com que se pode entrar com informações.

O tamanho do reservatório (buffer) é crítico e precisa ser conhecido pelo usuário — o sistema operacional CP/M, por exemplo, armazena um único toque; muitos microcomputadores acumulam dez toques ou mais.

Os acumuladores de toques, no entanto, também causam problemas. Um usuário experiente, trabalhando com sistema dirigido por menus, pode saber de antemão as escolhas de menu que fará. Por exemplo, 2, no menu principal, 5, no seguinte, e depois 3, 4 e 6. Graças a sua familiaridade com o sistema, ele digita essas escolhas com grande velocidade. Com um acumulador de dez caracteres, o usuário chega onde quer porque os toques são "lembrados" na seqüência correta. Com um acumulador de um caractere, o tempo necessário para expor o segundo menu pode ser mais longo do que o gasto para digitar a seqüência. Nesse caso, em vez de selecionar a alternativa número 5 desse menu, depois a 3, e assim por diante, apenas a alternativa número 6 é selecionada (porque esse é o único caractere guardado no acumulador) e o sistema pára nesse ponto.

Acumuladores grandes, contudo, também apresentam alguns inconvenientes. Um programa dirigido por menus que demore um tempo demasiado longo para reagir à digitação no teclado (isso ocorre se a escolha implica a leitura de um arquivo) pode fazer com que o usuário pense que nada está acontecendo. Sua reação natural será tentar novamente a última alternativa introduzida e pressionar várias teclas, até obter uma resposta. Muitas vezes, o programa tenta processar todos esses caracteres errados que ficam guardados no acumulador, oferecendo resultados surpreendentes.

A "limpeza", isto é, a eliminação dos registros da memória do computador a fim de liberar espaço para o trabalho é outra fonte de problemas. Tem-se a impressão de que o programa está parado por longo período, durante o qual podem ser tentadas ações corretivas. A limpeza, às vezes, causa problemas em programas grandes que lidam muito com strings (variáveis alfanuméricas).

Sistema CP/M

Algumas versões do BASIC permitem ao programador forçar uma limpeza; convém fazer isso a intervalos freqüentes e, nesse caso, o usuário deverá receber uma mensagem "por favor, es-

pere'' enquanto o computador estiver aparentemente inativo, mas procedendo à limpeza.

A maneira como a linguagem de programação lida com a entrada e a saída influencia o projeto da interface entre o computador e o usuário. Os bons recursos do BASIC para se lidar com strings permitem usá-las com mais sofisticação no diálogo entre homem e computador do que linguagens como PASCAL. Os BASICs dotados de comandos residentes para endereçamento do cursor proporcionam melhor diagramação *(layout)* de tela do que os outros. O mesmo se aplica para BASICs com comandos de gráficos.

O BASIC é bem suprido com comandos de entrada/saída — INPUT e PRINT são apropriados para programas simples. Mas, para controle efetivo da entrada (o tipo necessário para a produção de formulários que contenham campos protegidos para a entrada, por exemplo), experimente os GET$, INKEY$, INPUT$() e comandos similares. PRINT USING é um comando extremamente versátil para a formatação da saída; seu valor é inestimável para alinhar pontos decimais e ajustar colunas de texto.

Sistema Micronet 800

A psicologia do usuário

O usuário é o elemento mais imprevisível em todo sistema homem/máquina. Como qualquer outro componente, no entanto, ele possui certas características de desempenho que devem ser entendidas antes de se projetar a interface.

Usuário e computador são ''processadores de informação''. Contudo, os seres humanos têm limitações inerentes quanto à quantidade de informações novas que podem guardar na ''memória funcional'' — calcula-se que, para a maior parte dos tipos de informação, o cérebro pode guardar por volta de sete itens diferentes simultaneamente. O tamanho desses itens depende de quão significativos ou bem estruturados sejam. Se a informação consistir em caracteres aleatórios, cada item terá um caractere e nada mais. Mas, se, ao contrário, os caracteres formam palavras comuns, cada item lembrado pode ser uma palavra inteira. À medida que se amplia a estrutura das informações, aumenta a capacidade do usuário de lembrá-las e usá-las.

Alguns procedimentos ajudam a estruturar as informações. Um método consiste em relacionar os dados com estruturas familiares — assim funciona a metáfora da ''escrivaninha''; ou seja, vários tipos de informação — relatórios, gráficos etc. — são visualizados simultaneamente em janelas na tela, como os papéis na mesa de um executivo. Da mesma forma, um programa de planilha financeira pode ser organizado de modo a parecer um livro, com páginas e índices. Outro método é treinar o usuário para entender

Sistema Macintosh

estruturas não familiares. Por meio de exemplos repetidos e da explicação dos tópicos, o próprio programa ensina ao usuário como estruturar a informação. O inconveniente disso está no tempo e esforço despendidos. Instruções detalhadas e telas de ''ajuda'' fornecem um tipo de treinamento em linha, mas isso é difícil de usar com eficiência. Por fim, a apresentação da informação em padrões reconhecíveis constitui outro auxílio para o usuário, empregando-se cores ou a própria diagramação para ressaltar a informação desejada. Para ter idéia de como isso funciona, considere as instruções coloridas usadas em programas de videotexto. Numa página, ou tela, típica, o ''cabeçalho'' e o ''rodapé'' são blocos da mesma cor; há apenas uma cor de fundo e o texto é exibido em outras duas cores, que se alternam nos parágrafos. As palavras-chaves são enfatizadas pelo uso de outra cor. O propósito disso é permitir ao usuário selecionar somente a informação necessária. As instruções coloridas podem, no entanto, ser confusas se usadas em demasia, e alguns testes indicam que as pessoas desperdiçam muito tempo lendo e relendo parágrafos para tentar entender o significado das mudanças de cores. Uma boa regra é: nunca use mais de quatro cores de uma só vez.

Três graus
Estas fotos de telas de sistemas operacionais de microcomputadores ilustram três graus de interação com o usuário. Na primeira, um novato está tentando se comunicar com o sistema operacional CP/M. Mas este não tem recursos de ajuda ao usuário e, assim, exige conhecimento prévio completo dos comandos. O segundo exemplo é de um sistema dirigido por menus — o menu inicial para o correio eletrônico Micronet 800 no BBC Micro. As opções são numeradas e o usuário faz sua opção entrando com o número apropriado. A tela não oferece muitas informações e, dessa forma, o usuário precisa entender as opções. A última foto mostra o sistema operacional do Macintosh, da Apple, que fornece indicações visuais e símbolos gráficos, assim como menus de simples e fácil compreensão.

SOLUÇÕES INTEGRADAS

Calculadoras portáteis e de mesa, máquinas de escrever e videogames, computadores — o nome Dismac se firma como uma solução nacional nos vários ramos da eletrônica.

A Dismac Industrial S.A. foi fundada em 1973, em Manaus, com o objetivo principal de conquistar uma fatia do mercado brasileiro de calculadoras eletrônicas, então dominado por equipamentos importados. Doze anos depois, 60% das calculadoras eletrônicas de mesa comercializadas no Brasil trazem a marca Dismac.

Esse desempenho decorreu, em boa parte, da vasta experiência empresarial de seu fundador, o austríaco Joseph Feder. Em 1947, ele criou, na Itália, a fábrica de máquinas de costura Elgin, que, quatro anos depois, instalaria sua filial no Brasil. Em 1969, a Elgin brasileira já vendia suas máquinas para toda a América Latina e começava a penetrar nos mercados da Alemanha, Grã-Bretanha, Estados Unidos e da própria Itália.

No final de 1972, Joseph Feder afastou-se da diretoria executiva da Elgin, passando a Dismac a mobilizar toda a sua atenção. Os resultados foram imediatos. Em 1973, surgiu a linha de calculadoras eletrônicas portáteis; em 1974, iniciou-se a produção de calculadoras eletrônicas de mesa, com doze dígitos de operação; em 1978, foram lançadas as caixas registradoras eletrônicas; em 1979, a Dismac tornou-se o maior exportador brasileiro de calculadoras eletrônicas e se preparava para novos empreendimentos, que culminaram, em 1981, com o D-8100, o primeiro micro fabricado no Brasil.

A versatilidade e o baixo preço em relação ao desempenho asseguraram rápida difusão ao D-8100. Trata-se de um microcomputador compatível com o Apple II, com 48 Kbytes de RAM e 12 Kbytes de ROM. Pode operar com televisor tanto em cores como em branco e preto ou com o monitor de vídeo; possui saída para áudio, interface para impressora e unidades de disco flexível de 5 1/4 polegadas. Em 1984, foram lançadas novas interfaces, permitindo a monitoração em vídeo colorido (placa PAL-M) e saída serial ou paralela para impressora.

Desenvolveu-se, a seguir, a linha de computadores. O Alfa 2000 e o Alfa 3000 são computadores profissionais de 64 Kbytes de RAM. O Alfa 2064 é um sistema multiusuário que permite a conexão de até oito terminais trabalhando em tempo real. A Dismac criou também uma série de pacotes aplicativos para a linha Alfa, possibilitando a utilização dos sistemas em qualquer escritório. Entre os aplicativos destacam-se os de folha de pagamento, faturamento, controle de estoques, contabilidade e crediário.

Em 1983, surgiram novos modelos de caixas registradoras eletrônicas, aptas a funcionar como terminais. Cada unidade pode operar em comunicação direta com o sistema central Dismac 3003 — um gerenciador da linha Alfa —, tornando-se um terminal automatizado de venda.

A busca de soluções integradas para facilitar o controle empresarial não impediu a Dismac de diversificar suas atividades. Ainda em 1983, foram lançados o Dismac Vídeo Game — que opera em cores no sistema PAL-M — e o Super Stick, um joystick de precisão, compatível com todos os aparelhos do sistema Atari.

Em 1985, foi a vez da Olympia Dismac OAT 1200, a máquina de escrever eletrônica brasileira de maior capacidade de armazenamento de dados do mercado: armazena até 1.000 caracteres em sua memória. Além disso, é a única a possuir memória partilhada (em até dez compartimentos), o que torna possível gravar, em separado, cabeçalhos, endereços e frases inteiras habitualmente utilizadas.

A Dismac respondia, em 1985, por mais de 30% do mercado nacional de caixas registradoras eletrônicas e por quase 20% do mercado de computadores. Na segunda metade dos anos 80, ela pretendia repetir esse desempenho favorável nos campos do lazer eletrônico e das máquinas de escrever ultra-sofisticadas.

Tecnologia própria
Com suas linhas de produção instaladas em Manaus, a Dismac é uma das empresas da área de informática que apresentam os mais altos índices de nacionalização de seus produtos.

DADOS A DISTÂNCIA

O tom de comando
Desenvolvido originalmente para conectar terminais distantes a computadores de grande porte, o modem converte (ou modula) dados digitais em tons acústicos para transmissão telefônica e, na recepção, demodula os tons acústicos de volta em sinais digitais. Os acopladores acústicos transmitem e recebem por meio do aparelho telefônico, enquanto os modems de conexão direta se ligam à linha telefônica.

"Telecomunicações" é o termo usado para descrever o processo de enviar informações (tanto de voz como de dados) pela rede telefônica. Nesse sentido, o modem revela-se um dispositivo de grande utilidade.

A conexão de seu micro a um modem lhe possibilita comunicar-se com outros computadores pela linha telefônica. Assim, muitas informações são acessadas. Você pode mandar cartas, que serão recebidas no instante em que forem transmitidas, intercambiar software com amigos distantes e mensagens com muitos outros usuários de microcomputadores, além de acessar computadores de grande porte.

A tecnologia de telecomunicação tem sua origem nos computadores de grande porte. Numa configuração típica, a CPU de um grande computador está numa sala especialmente construída, com ar condicionado, e se conecta a terminais espalhados por todo o prédio. Um terminal compõe-se apenas de uma tela e um teclado ligados ao computador central por meio de um cabo serial. Dessa forma, o usuário pode utilizar um terminal num extremo do prédio e acessar o computador no outro extremo.

Uma ligação por cabo entre o computador e o terminal funciona bem para distâncias relativamente curtas — no máximo, algumas centenas de metros —, mas a deterioração do sinal exclui o uso desse sistema para distâncias maiores. Para resolver esse problema, desenvolveu-se o modem (*mo*dulador/*dem*odulador).

Esse dispositivo permite que dados sejam transmitidos por uma linha telefônica comum. Ele transforma os sinais digitais em tons acústicos (sinais analógicos) de freqüência e volume apropriados para a transmissão pela rede telefônica. O processo é conhecido por "modulação". O modem do computador que recebe a transmissão reconverte esses tons acústicos de volta em sinais digitais que passam para o computador receptor. É a demodulação. Usa-se um

sinal constante (chamado "sinal de portadora") como referência, enquanto o dado é transmitido na onda senóide (modulada).

Como resultado desse processo, acessam-se computadores distantes quase como se estivessem diretamente conectados ao terminal. Os terminais dedicados raramente têm poder próprio de processamento, sendo então chamados de "terminais não-inteligentes". É possível empregar um microcomputador como terminal, simplesmente usando-se software de comunicação apropriado. Contudo, como o micro tem sua própria capacidade de processamento, é chamado de "terminal inteligente".

Existem dois tipos de modems: o acoplador acústico e o modem de conexão direta. O acoplador acústico é mais simples: tem dois bocais de borracha em que se coloca o gancho do telefone. Os sinais acústicos são transmitidos pelo bocal e recebidos pelo fone. O modem acústico é, dos dois tipos, o mais conveniente para sistemas portáteis e microcomputadores, pois pode ser usado com qualquer telefone. Há mesmo quem use modems acionados a pilha com computadores portáteis, fazendo-se a ligação a partir de telefones públicos. Por outro lado, os modems acústicos estão sujeitos a interferência (ruídos do ambiente) e, por isso, não são tão confiáveis. Os modems de conexão direta têm quatro saídas: duas para a linha telefônica e duas para os fios do telefone propriamente dito. Além de serem mais confiáveis que seus equivalentes acústicos, os modems de conexão direta em geral oferecem mais recursos.

No Brasil, os modems são aprovados pela Embratel e sua fabricação segue as normas internacionais de comunicação. Essas duas condições garantem que os modems funcionem de acordo com as normas telefônicas, evitando ruídos ou perdas de sinais.

Alguns modems oferecem recursos como "auto-resposta" e "autodiscagem". Os modems com auto-resposta possibilitam o atendimento automático da chamada. Quando atendem o telefone e detectam a portadora do modem da outra extremidade, passam o controle para o computador. Se a portadora não for detectada, o modem de auto-resposta desliga o telefone. Os modems de autodiscagem recebem um número do computador e automaticamente o discam. Dessa forma, em resposta a um nome digitado pelo usuário, o software do computador consulta o número do telefone num banco de dados e faz com que o modem disque tal número.

Mede-se a velocidade de transmissão de dados entre dois dispositivos em bauds. Normalmente, considera-se tal velocidade como o número de bits transmitidos por segundo, embora esta não seja uma definição precisa.

As velocidades de transmissão ou taxas de bauds mais comuns para dispositivos de comunicação são 300, 1.200 e 1.200/75 (esta se refere a dados recebidos a 1.200 bauds e transmitidos a 75). A maioria dos micros grava programas em cassete com uma velocidade dentro da mesma faixa (entre 300 e 1.200 bauds).

Vários padrões de comunicação são utilizados em computadores de pequeno e de grande por-

Transmissão e verificação

Bit de paridade
Transmitem-se os dados pela linha telefônica, em código binário correspondente a cada caractere, como uma seqüência de tons acústicos. A freqüência do 0 binário fica um pouco abaixo do tom de referência, ou sinal de portadora, ao passo que o 1 binário tem freqüência um pouco acima. Para se detectar erro, geram-se os bits de paridade.
Por exemplo: aqui, o caractere transmitido é o A—65 em código ASCII; 01000001 em binário. Há um número par de dígitos 1 neste código e a paridade é ímpar, o que significa que a soma dos dígitos 1 do caractere deve ter um número ímpar de uns; o bit de paridade é, assim, estabelecido como 1. Em seguida, há o bit de fim (stop), sinalizando o término da palavra transmitida. Essa palavra tem 8 bits, mas exige 10 bits para sua transmissão.

te. Podem variar a velocidade, o tamanho do bloco de informação transmitido, os bits de paridade e até mesmo o sistema telefônico.

65 = 01000001

Por causa das diferenças entre os sistemas telefônicos do Brasil e dos Estados Unidos, por exemplo, os dois países usam padrões de sinais telefônicos diferentes. O padrão brasileiro é conhecido como CCITT (ou V21), enquanto o equivalente americano é o Bell (nome derivado da companhia telefônica Bell).

Como o sistema telefônico só pode diferenciar um número limitado de freqüências com algum grau de confiabilidade, os dados são transmitidos em forma binária. Para tanto, cada caractere no código ASCII é traduzido para sua forma binária. A letra A converte-se do código 65 (ASCII) para o 01000001 (binário). Os dígitos 1 são representados por uma freqüência "alta", pouco acima do sinal de portadora; uma freqüência "baixa", pouco abaixo do sinal de portadora, representa os dígitos 0.

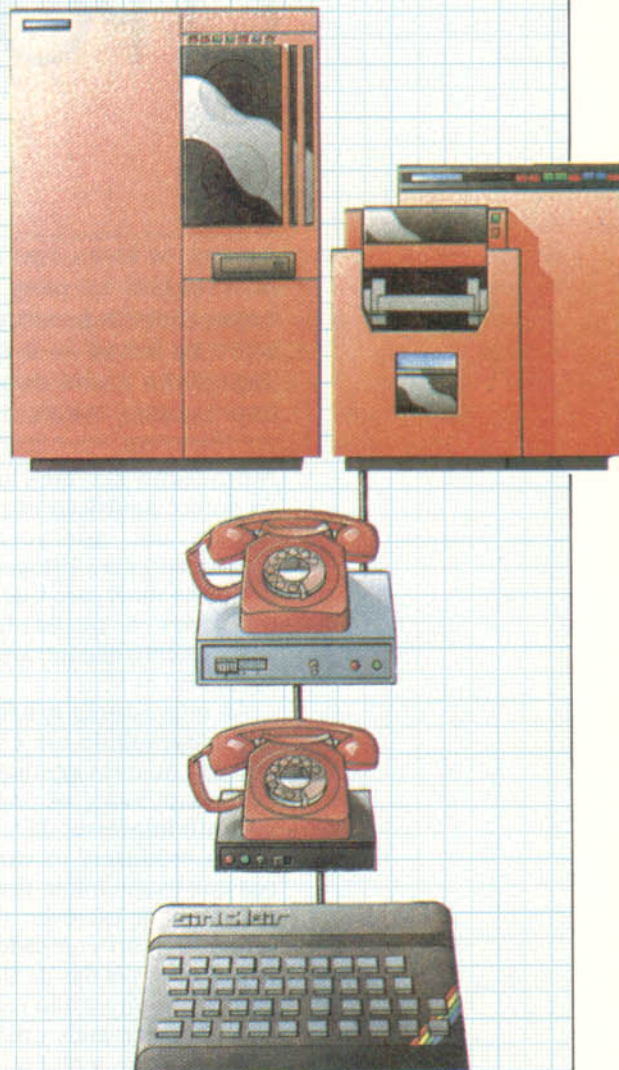
O computador que recebe a informação precisa identificar onde termina um caractere e começa o próximo. Usam-se, portanto, bits de início (start) e de fim (stop). Esses dois bits são delimitadores da palavra (conjunto de códigos ASCII) transmitida e há convenções que os estabelecem. Quando recebe um bit de fim, por exemplo, o computador decodifica os bits precedentes. O software que controla toda a comunicação é o protocolo. Sua função é disciplinar o meio de comunicação. Ele decodifica a palavra após receber o bit de fim (stop) e codifica os dados recebidos e transmitidos para a transmissão e verificação da paridade. O protocolo mais comum para comunicação em ASCII é o XON/XOFF, de 8 bits de informação seguidos por 1 bit de fim. Um esquema alternativo é o de 7 bits de dados e 2 bits de fim. Outros sistemas usam tanto 1 bit de início quanto 1 de fim — a mudança indica o fim do caractere.

Uma vez que nenhuma comunicação é totalmente confiável, precisamos de uma forma para verificar se houve erro. A solução mais simples, conhecida como "verificação da paridade", consiste em contar o número de bits altos (1) e adicionar 1 bit extra de paridade para fazer com que o total de bits altos seja ímpar (num sistema de paridade ímpar) ou par (num sistema de paridade par). Por exemplo, voltando à letra A, o número total de bits altos em 01000001 é 2 (um número par). Dessa forma, para tornar a paridade ímpar, o bit de paridade também terá de ser alto. Já no caso da letra C — traduzida como 01000011 em código binário —, o bit de paridade terá de ser baixo.

A verificação de paridade é pouco sofisticada: detecta um erro num único bit, mas não detecta dois erros no mesmo caractere, estando a paridade certa. É, contudo, simples e útil para as aplicações em que a precisão não é vital.

Fluxo de informação

A taxa de bauds, assim denominada em homenagem a Émile Baudot, pioneiro francês da transmissão de dados, é uma medida da velocidade do fluxo de informações entre dispositivos de comunicação. A taxa de baud é 1 se a informação for transmitida a 1 bit por segundo. Tipicamente, dispositivos que se comunicam por modem usam taxas que variam entre 300 e 9.600 bauds.

No estabelecimento da taxa de bauds, devem-se considerar os bits de início, de fim e de paridade. Por exemplo: num sistema em que há verificação de paridade e que usa 8 bits de dados e 1 bit de fim para cada caractere, apenas 8 em cada 10 bits transmitidos contêm informações úteis. Assim, numa velocidade de transmissão de 300 bauds, o verdadeiro número de bits por segundo é 240 (80% de 300).

Registre

Linhas Apple e TRS-80

TÚNEL COLORIDO

Este programa em BASIC produz belo efeito visual na tela: um túnel que se afunila e muda de forma, produzindo impressão tridimensional. As cores mudam aleatoriamente, intensificando o efeito.

```
5 REM *** TUNEL COLORIDO ***
10 GR : HOME
20 FOR I =1 TO 5
30 FOR N = 0 TO 39
40 W = INT ( RND (1) * 15 )
50 COLOR = W
60 VLIN 0,39 AT N
70 VLIN 0,39 AT 39 - N
80 HLIN 0,39 AT N
90 HLIN 0,39 AT 39 - N
100 NEXT N, I
110 END
```

SINAIS

Digital:
Forma de onda quadrada, na qual existem apenas dois níveis: 0 V e +5 V. É o sinal utilizado internamente nos computadores.

Analógico:
Pode ser descrito por uma senóide, em que é aceita qualquer variação, em freqüência, amplitude ou fase.

O PODER DA FLOR

As impressoras tipo margarida apresentam funções práticas — como espaçamento proporcional entre as letras — e qualidade de impressão muito superior à das impressoras matriciais.

À primeira vista, uma impressora tipo margarida parece aquisição estranha para o proprietário de um microcomputador. Além de inadequada para listagens de programas, é lenta e custa mais caro que uma impressora matricial. Mas a qualidade de impressão desse equipamento encontra-se muito à frente de seus similares.

As impressoras matriciais formam os caracteres imprimindo um conjunto de pontos: não importa quantas agulhas sejam usadas na cabeça de impressão, os pontos sempre aparecerão no texto impresso. Já as impressoras tipo margarida produzem os caracteres por meio de tipos que batem numa fita entintada — como nas máquinas de escrever. Os tipos, um para cada caractere, são colocados em círculo, assemelhando-se a pétalas de flor — daí o nome "margarida". A impressão resultante é de leitura mais fácil e parece mais profissional.

No entanto, a alta qualidade tem sua contrapartida: a impressão apresenta a grande desvantagem da pequena velocidade. Para imprimir um caractere, a margarida gira até a "pétala" desejada ficar no alto; daí, o martelo de impressão bate no tipo e o carro se move para produzir outro caractere.

Comparando esse processo com o da impressora matricial, que imprime os pontos à medida que o carro anda de um lado para outro do papel, entende-se a razão da diferença de velocidade. Uma impressora tipo margarida padrão imprime cerca de vinte caracteres por segundo (cps) — há mais velozes, de oitenta cps, mas custam muito mais caro; uma impressora matricial Epson FX-80, por exemplo, custa o mesmo, mas tem velocidade de 160 cps.

Duas funções extras permitem aumentar a velocidade das impressoras tanto matriciais como do tipo margarida: impressão bidirecional e procura lógica. Bidirecional significa apenas que uma linha do texto é impressa da esquerda para a direita enquanto a linha seguinte o é da direita para a esquerda. A impressora não precisa esperar o carro voltar e, assim, pode imprimir com maior rapidez. Procura lógica significa que o carro pula espaços para chegar à palavra seguinte do texto — impressoras menos sofisticadas levam o mesmo tempo para imprimir um espaço ou qualquer caractere.

As formas dos caracteres de uma impressora matricial estão armazenadas na ROM da impressora; os caracteres das impressoras tipo margarida estão armazenados na própria margarida, em forma de tipos físicos. Cada método tem suas vantagens: com a matriz de pontos, as formas dos caracteres podem ser redefinidas enviando-se códigos adequados para a impressora a partir do micro. Na impressora tipo margarida, esse processo é muito mais fácil: basta trocar a margarida por outra, diferente.

As margaridas possuem grande variedade de modelos de tipos e pitches; o modelo de tipo refere-se ao desenho dos caracteres, enquanto o pitch diz respeito à largura do caractere. Os modelos mais comuns de tipo são: courier, romano, gótico e itálico. O pitch, em geral, é de dez ou doze caracteres por polegada.

O único problema com os diversos modelos é que poucos têm os caracteres exatamente iguais aos usados por um micro. Por exemplo, o caractere especial # sai, em geral, impresso com o símbolo do cifrão ($), um colchete ([) aparece no papel como a fração 1/2. Em muitos modelos, não se pode distinguir o número 0 (zero) da letra O maiúscula. Da mesma forma, a letra l minúscula pode ser confundida com o número 1. Enquanto as impressoras mais caras possuem margaridas de 127 caracteres, a maioria imprime apenas 92 ou 96 caracteres, e são estas as que costumam apresentar esse tipo de problema.

Imagine como se torna muito mais difícil depurar um programa quando é impossível distinguir o número 1 da letra l. Por isso, e também por sua lentidão, não se recomenda a aquisição de uma impressora tipo margarida para quem usa o micro principalmente em programação.

Efeitos especiais

À semelhança das matriciais, as impressoras tipo margarida podem ser programadas para produzir efeitos especiais, usando os mesmos métodos. O número desses efeitos, no entanto, é limitado. Por exemplo, alguns modelos dispõem de códigos especiais para ativar e desativar o sublinhado automático, marcar um ponto tabulado, determinar a margem esquerda e empurrar o papel meia linha para cima ou para baixo — permitindo a impressão de caracteres subscritos ou superescritos.

As impressoras tipo margarida também contam com recursos que possibilitam efeitos equi-

valentes ao texto em negrito de uma impressora matricial, em que cada caractere é reimpresso quatro vezes, para ficar destacado. Esse tipo de escrita, porém, apresenta a desvantagem de reduzir bastante a velocidade de impressão.

Em alguns modelos é possível variar a distância entre os caracteres e o espaço entre as linhas. Nesse caso, a impressora serve para criar imagens gráficas e dumps de tela. Se um pixel de tela está "ligado", a margarida imprime um ponto; se estiver "desligado", imprime um espaço. Reduzindo a distância entre os caracteres, pode-se passar uma linha horizontal inteira da tela para o papel. Da mesma forma, reduz-se o intervalo das linhas para não deixar espaço entre elas. Contudo, esse processo é muito lento.

Uma característica não encontrada nas impressoras matriciais é a capacidade de centralização automática dos títulos. Outro aspecto favorável é a tabulação decimal, que alinha as vírgulas decimais de uma lista de números, facilitando a adição de valores monetários.

Espaçamento estético

As impressoras tipo margarida mais caras realizam espaçamento proporcional. O pitch não é mais padronizado, com dez ou doze caracteres por polegada, mas varia conforme a largura do caractere impresso. Por exemplo, o caractere w é muito mais largo que o i; com espaçamento proporcional, o carro não avança tanto para um i quanto avançaria para um w. Isso significa que os caracteres em linhas sucessivas do texto não ficam exatamente uns embaixo dos outros, e o aspecto geral é mais agradável aos olhos — como as linhas desta página. Na maioria das vezes, precisa-se de uma margarida especial para usar corretamente essa função, mas ela é ligada e desligada por meio de códigos — como em qualquer outro efeito da impressora.

Num computador comercial, o uso desse tipo de impressora justifica-se plenamente, o que não ocorre no caso dos usuários de micros domésticos. Uma alternativa é a adaptação de uma máquina de escrever eletrônica, menos cara que a impressora tipo margarida, embora use o mesmo método de impressão.

Até há pouco tempo, as máquinas de escrever eletrônicas não podiam ser ligadas a um micro, pois não dispunham de interface ou saída serial RS232. Contudo, alguns modelos lançados na metade da década de 80 já saem da fábrica com interface embutida, ou permitem adaptar um kit de interface. Além do preço mais acessível, a vantagem de se utilizar essas máquinas eletrônicas é que elas continuam funcionando como máquinas comuns. Assim, você terá uma máquina para escrever cartas personalizadas ou endereçar envelopes e, ao mesmo tempo, uma impressora para o processamento de textos. No Brasil, dada a inexistência de modelos comerciais de impressoras tipo margarida, a única alternativa à importação é adaptar uma das máquinas eletrônicas existentes como, por exemplo, a Olivetti ET 121, a Olympia Dismac OAT 1200 etc.

Recursos da margarida

AMOSTRA DE IMPRESSÃO COM 10 CARACTERES POR POLEGADA

Amostra de impressão com 10 caracteres por polegada

AMOSTRA DE IMPRESSÃO COM 12 CARACTERES POR POLEGADA

Amostra de impressão com 12 caracteres por polegada

AMOSTRA DE IMPRESSÃO COM 15 CARACTERES POR POLEGADA

Amostra de impressão com 15 caracteres por polegada

O USO DE NEGRITO DESTACA PARTES DO TEXTO

O uso de negrito destaca partes do texto

TEXTO CENTRALIZADO

Texto Centralizado

Com espaçamento proporcional, obtém-se por exemplo, w com a mesma largura de i :

wwwwwwwwwwwwwwww

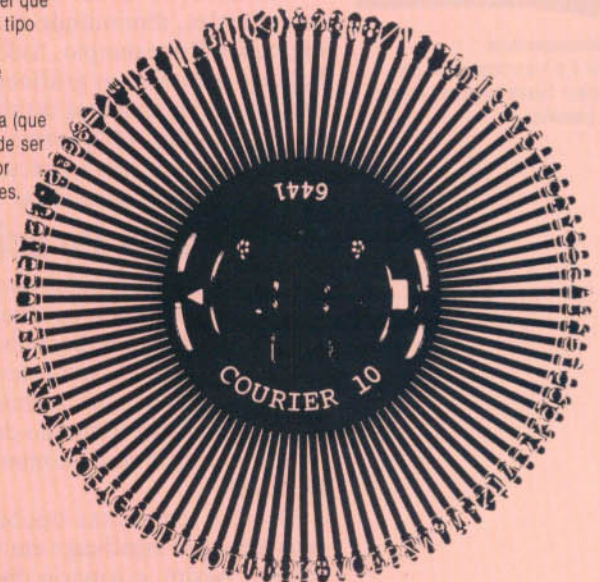
iiiiiiiiiiiiiii

Trocando-se as margaridas obtemos diferentes tipos de caracteres

0123456789

0123456789

0123456789

Mais cara e menos flexível que a matricial, a impressora tipo margarida imprime com qualidade de máquina de escrever e espaçamento proporcional. A margarida (que se parece com a flor) pode ser facilmente substituída por outra, com tipos diferentes.

1-2-3: MACROCOMANDOS

Ampliando os recursos do Lotus 1-2-3, os macrocomandos permitem programar esse software integrado de modo que ele adquire características de uma verdadeira linguagem.

As planilhas projetadas para micros de uso pessoal sofrem restrições devido à pequena quantidade de RAM disponível para o programa e para os modelos financeiros desenvolvidos. Programas similares, escritos para o IBM PC, o Apricot da ACT e outros equipamentos profissionais, podem dispor de memória residente extensa e maior velocidade de processamento.

Em conseqüência, alguns dos softwares mais inovadores — a exemplo do Lotus 1-2-3, que integra planilha, banco de dados e gerador de gráficos — só rodam em máquinas caras, daqueles tipos.

No entanto, o 1-2-3 oferece ao usuário recursos úteis, como a possibilidade de programar com os macrocomandos.

O macro é uma seqüência de comandos freqüentemente utilizada. A representação gráfica desses comandos é armazenada na planilha, e recebe um nome especial, formado por uma barra invertida (\) e uma letra. Para executar a seqüência toda, basta pressionar [ALT] (no teclado do IBM PC e compatíveis) e mais a letra escolhida para o macro. Assim programado, o Lotus 1-2-3 roda sozinho, podendo executar um número teoricamente infinito de comandos.

Dessa forma, tarefas repetitivas podem ser automatizadas, diminuindo o tempo e os erros de digitação. Por exemplo, todos os cálculos de um balanço mensal, e os gráficos e relatórios que ele produz, são planejados detalhadamente, comando a comando, sob a forma de um ou mais macros. Feito isso, para executar a rotina mensal completa, basta pressionar algumas teclas.

A tela do Lotus
No Lotus, para trazer o menu principal à tela, basta teclar [/], como no Visicalc.

Macroeconomia
Este é o macrocomando para nomear faixas tal como aparece na planilha.

O programa em ação

As linhas são designadas por letras e as colunas, numeradas. Dirigido por menus, o 1-2-3 mostra na tela os comandos disponíveis, quando se pressiona a tecla [/]. A partir desse ponto, as opções são selecionadas com a tecla correspondente à primeira letra do comando desejado, ou posicionando o cursor nesse comando e pressionando [Return].

O 1-2-3 tem tantas opções de comando que seus menus se ramificam em vários níveis de submenus. Assim, o usuário dispõe do Lotus 1-2-3 para realizar, literalmente, centenas de tarefas; mas isso também significa que algumas operações exigem muitos toques no teclado, como mostra o diagrama.

O 1-2-3 permite dar nome a uma célula ou faixa de células contíguas. Para operar com a faixa nomeada — incluí-la numa fórmula, por exemplo —, o nome é usado no lugar do endereço da célula, como segue:

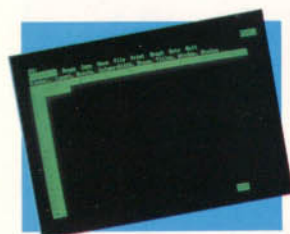
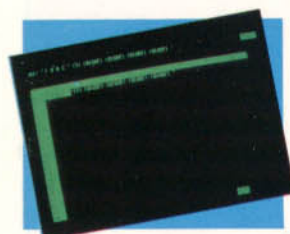
A3 – B3 = C3	VENDAS – CUSTO = LUCRO
(endereços de células)	(nomes de células)

Dar nomes às regiões simplifica e acelera o processo de trabalhar com uma planilha extensa. Contudo, nomear um grupo de quatro células no 1-2-3 requer os seguintes comandos:

/ — mostra o menu principal.
R(ange) — diz ao programa que você vai realizar uma operação em número restrito de células e não na planilha inteira.
N(ame) — indica que você vai dar um nome à faixa especificada de células.
C(reate) — prepara o 1-2-3 para aceitar um nome e adotá-lo.
Digite o nome a ser adotado — CUSTO, por exemplo. Tecle [Return] para aceitar a célula indicada pelo cursor como começo da faixa. Mova o cursor quatro espaços para a direita. Tecle [Return] para aceitar a célula indicada pelo cursor como o fim da faixa.

Esse processo exige dez toques no teclado, sem contar a digitação do nome da faixa. Para reduzir o número de toques, podemos montar um macrocomando para nomear faixas automaticamente. Começa-se por fixar a localização dos macros na planilha, escolhendo uma seção de células para isso.

Tal escolha exige cuidados especiais, por dois motivos. Em primeiro lugar, as células devem ocupar uma região segura da planilha, uma área em que nunca serão colocados dados. Em segundo, o 1-2-3 só aloca na memória as células indicadas pelo cursor; assim, os espaços vazios entre as células de dados ocuparão lugar na memória.

Se o macro for colocado muito à direita do resto da planilha, inúmeros Kbytes de memória utilizável poderão ser consumidos por células vazias. Por essa razão, muitas vezes é preferível colocar os macros na extremidade esquerda da planilha. Considere um macro construído na coluna A. Se o número de comandos for excessivo e não couber numa célula, o macro poderá ficar contido em linhas sucessivas da mesma coluna. Posicionando o cursor na célula A1, digitam-se as iniciais dos comandos necessários, precedidas de um apóstrofo:

'/ R N C

Nesse ponto, o 1-2-3 espera pela entrada do nome escolhido para a faixa. Digitando um ponto de interrogação entre chaves, introduz-se uma pausa. O programa aguarda até que o usuário pressione [Return], para então prosseguir com o macro. Utilizam-se palavras entre chaves, quando se trata de uma ação que não pode ser indicada por tecla específica, como os movimentos do cursor (designados por uma palavra de direção entre chaves). A indicação dos returns se faz por um til. O macro deve agora comandar o cursor para mover-se quatro células para a direita (right), cobrindo a faixa designada como "CUSTO". Seu formato ficará assim:

'/ R N C {?} ~ {right} {right} {right} {right} ~

Depois de dar entrada a todos os comandos necessários para nomear uma região, o corpo do macro está pronto.

O próximo passo consiste em nomear a região onde está armazenado o próprio macro, o que será realizado com uma tecla — por exemplo, [N], de nomear. Será necessário digitar novamente todos os comandos, pois o macro ainda não foi nomeado nem armazenado. Coloca-se, então, o cursor na célula A1 e digita-se:

[/][R][N][C][\][N][Return][Return]

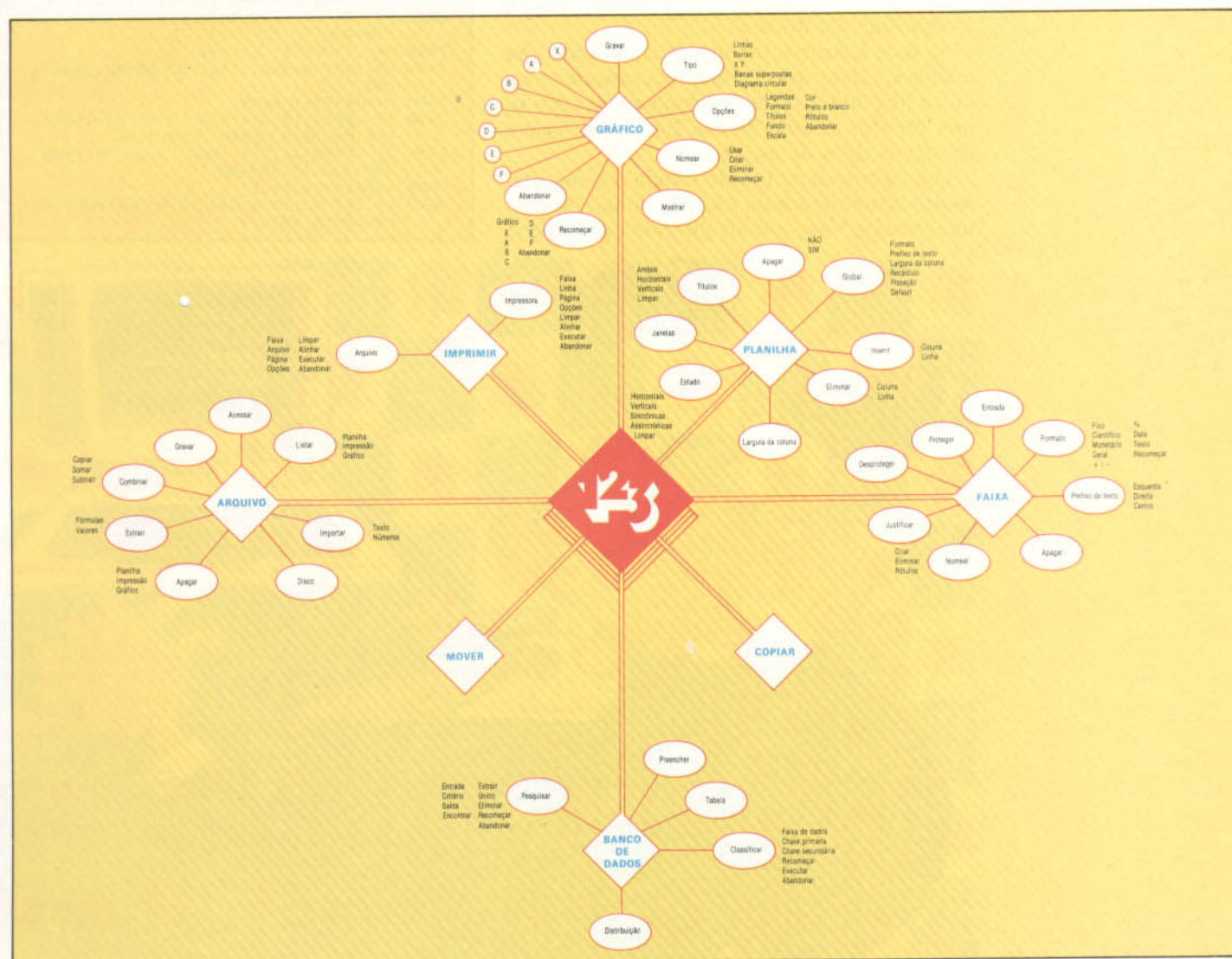
O caractere \ é usado para indicar que a tecla [ALT] deve ser pressionada; assim [\][N], o nome do macro, significa, para o 1-2-3, [ALT]+[N]. Agora que a região foi nomeada, todas as vezes que se pressionarem [ALT] e [N] simultaneamente, a seqüência de comandos armazenada será ativada. Desse momento em diante, para nomear uma faixa de quatro células, basta digitar:

[ALT][N] (nome da faixa) [Return]

Ou seja, há um grande avanço em relação ao procedimento inicial.

Esse exemplo, embora útil, é apenas uma pequena amostra do potencial dos macrocomandos.

O princípio dos macrocomandos permite que o usuário utilize aplicativos quase da mesma forma como alguém escreveria um programa BASIC ou PASCAL. Embora o Lotus 1-2-3 se limite, por enquanto, aos equipamentos de 16 bits, o conceito dos macros logo estará disponível em softwares destinados a equipamentos de uso pessoal.

A árvore de comandos Lotus
O Lotus tem vários níveis de menus. Estes comandos estão incorporados ao programa, enquanto outros podem ser definidos pelo usuário por meio dos macros.

PSIQUIATRA ELETRÔNICO

Simulando uma sessão de psicoterapia, o programa aqui analisado ilustra técnicas de manipulação de strings e dá uma idéia do grande desafio que representa a análise computacional da linguagem humana.

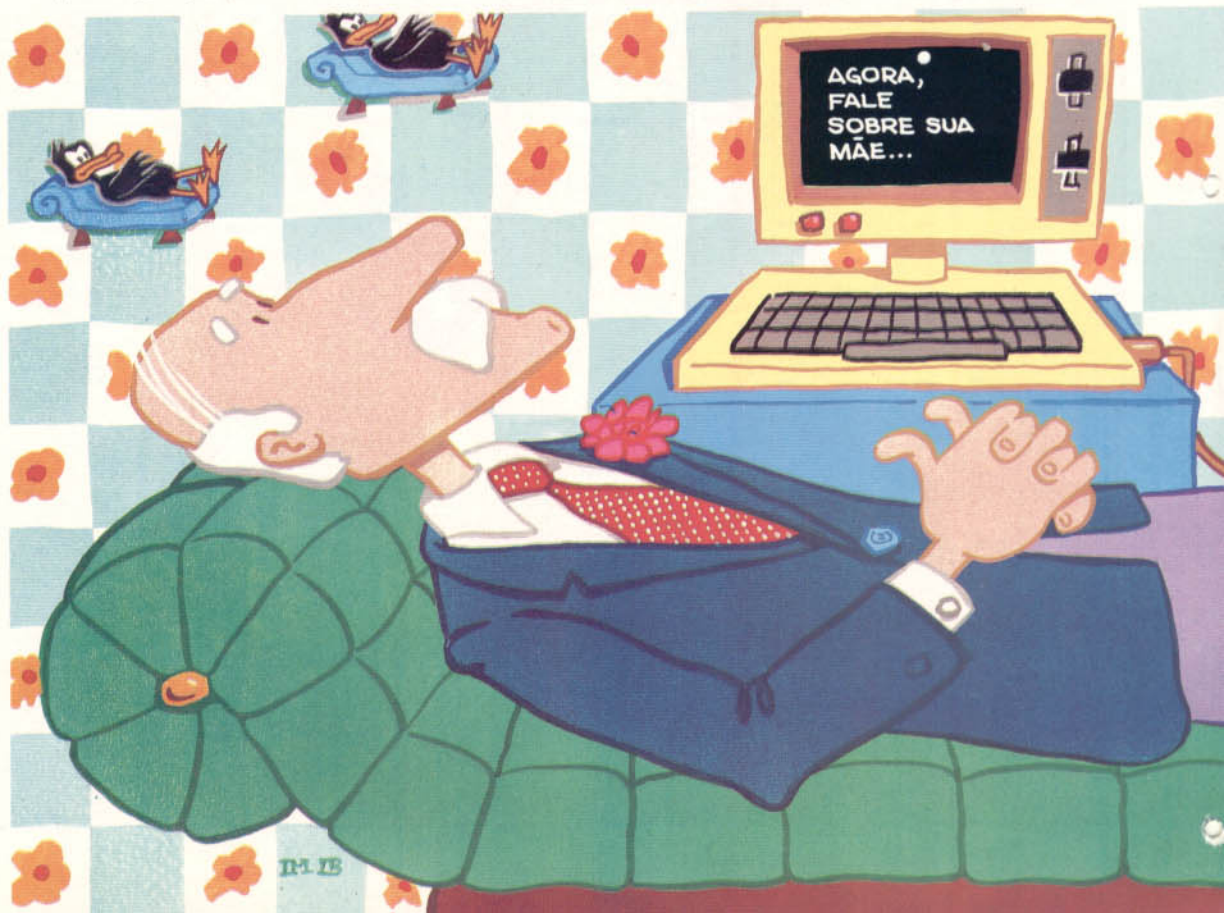
Um dos pioneiros na computação, Alan Turing, propôs um bom teste de inteligência artificial: se a máquina puder conversar com o ser humano sem que ele perceba que está dialogando com uma máquina, esta será "inteligente".

Até agora, nenhuma máquina passou nesse teste, embora vários programas aproximem-se da inteligência artificial. Não chegam a trocar opiniões com a pessoa, mas a estimulam a falar e a organizar seu discurso.

Se você refletir sobre uma boa conversa que manteve, talvez conclua que foi praticamente o único que falou. A outra pessoa apenas deu sinais sonoros de estar atenta. Esse é o princípio de alguns tipos de psicoterapia.

O programa aqui listado simula o comportamento de um terapeuta não direcional, ou seja, um psicólogo que jamais direciona a conversa

Análise

```
100  REM ********************************
110  REM *        PARA LINHA APPLE      *
110  REM *            ANALISE           *
120  REM ********************************
140  GOSUB 2000
200  FOR L=1 TO 2 * LT STEP 2
300  PRINT O$:INPUT I$
400  GOSUB 3000
500  NEXT L
1000 PRINT TAB(5);"---A SESSAO TERMINOU-
     --"
1100 PRINT"--DIGITE QUALQUER TECLA PARA
     CONTINUAR--"
1200 GET A$:IF A$="" THEN 1200
1300 GOSUB 5000
1900 END
1990 REM ********************************
2000 REM **          INICIALIZA       **
2010 REM ********************************
2050 LT=20:AN=10:T9=500:EX=2*LT
2100 DIM R$(AN):DIM H$(2*LT)
2200 DATA "SIM...","CONTE MAIS...","CONT
     INUE...","E...","ENTAO..."
2250 DATA "ISTO E IMPORTANTE...","POR QUE
     ISTO E IMPORTANTE PARA VOCE ?","EXP
     LIQUE MELHOR"
2300 DATA "POR QUE VOCE PENSA ASSIM ?"
2350 DATA "COMO ISTO AFETA VOCE ?"
2500 FOR K=1 TO AN:READ R$(K):NEXT K
2600 HOME:O$="OLA! QUAL E' O PROBLEMA ?"
2950 RETURN
2995 REM ********************************
3000 REM **           ANALISA         **
3010 REM ********************************
3100 IF I$="ATE LOGO" THEN EX=L-1:L=2*LT
     :RETURN
3200 H$(L)=O$:H$(L+1)=I$
3300 R9=INT(AN*RND(1)+1):IF R9=R0 THEN 3
     300
3400 O$=R$(R9):R0=R9
3420 IF LEFT$(I$,3)="SIM" THEN O$="VOCE
     TEM CERTEZA MESMO ?":RETURN
3450 IF LEFT$(I$,3)= "NAO" THEN O$="POR
     QUE NAO ?":RETURN
3500 Z$=RIGHT$(I$,1)
3520 IF Z$="?" THEN O$= "POR QUE VOCE ME
     PERGUNTA ISTO ?"
```

e sim a incita com respostas como: "Mas é importante mesmo?" e "Fale mais sobre isso".

O programa mantém um registro da conversa e o mostra depois de certo número de sentenças, ou tão logo você digite "ATE LOGO".

Quando desenvolvido em direção à pseudo-inteligência, faz com que o computador analise as entradas do usuário e escolha a resposta apropriada. Em nosso exemplo, ele faz isso já na linha 3100, testando para ver se é "ATE LOGO".

Pode-se estender essa análise verificando se há as palavras "sim" e "não" e questioná-las. Alguma coisa simples como "POR QUE SIM?" ou "POR QUE NAO?" já faria efeito.

Dá bom resultado, também, conferir se o usuário está repetindo uma resposta e, se estiver, responder de acordo ou encerrar a sessão. Pode-se ainda testar se a resposta acaba em ponto de interrogação ou de exclamação, e responder: "POR QUE VOCE ME PERGUNTA ISTO?" ou "POR QUE ISTO O IMPORTUNA?". Essas estratégias são eficazes e não exigem muito tempo de processamento nem compreensão da resposta do usuário.

```
3550 IF Z$="!" THEN O$="POR QUE ISTO O IM
     PORTUNA ?"
3600 COM = LEN (I$)
3610 IF COM > 15 THEN GOSUB 4000
3950 RETURN
3990 REM ********************************
4000 REM **        FRASE MAIS LONGA      **
4010 REM ********************************
4050 W=1:H=1:WL=1:S=1
4100 I$=I$+" ":L9=LEN(I$)
4120 FOR C=1 TO L9:FOR P=C TO L9
4150 Z$=MID$(I$,P,1)
4170 IF Z$=" " THEN W=P-C:H=C:C=P:P=L9
4200 NEXT P
4220 IF W>WL THEN WL=W:S=H
4250 NEXT C
4270 O$="O QUE SIGNIFICA PARA VOCE " +
     CHR$(13) + "'" + MID$ (I$,S,WL) +
     "' ?": RETURN
4990 REM ********************************
5000 REM **          RELATORIO           **
5010 REM ********************************
5050 HOME:PRINT TAB(10);"RELATORIO"
5100 FOR K=1 TO EX STEP 2
5150 PRINT TAB(5);H$(K)
5200 PRINT H$(K+1)
5300 FOR D=1 TO T9:NEXT D
5400 NEXT K
5900 RETURN
```

Podemos, ainda, criar uma tabela de palavras-chaves e procurar respostas para elas. A escolha das palavras-chaves e das respostas depende do tipo de conversa que se espera ter.

O inconveniente desse método — e de todos os métodos de pesquisa de texto — é o tempo que demanda para analisar mesmo uma frase curta. A baixa velocidade do BASIC é o fator limitante; qualquer análise séria, no caso, requer um programa escrito em código de máquina.

No entanto, essas demoras são perfeitamente toleráveis, e mesmo métodos simples podem levar a resultados surpreendentes, além de exemplificar os problemas que as máquinas de quinta geração terão de solucionar para compreender todas as sutilezas da linguagem humana, subjetiva e não padronizada.

Se pretendemos analisar as palavras do usuário, há muitas abordagens a serem seguidas. A mais interessante é a análise sintática — a redução de uma sentença a seus componentes, como sujeito, verbo e complemento. Isso exige um corpo de regras sintáticas e gramaticais e tabelas de pronomes, preposições, conjunções, flexões etc.

Seria mais fácil escolher as frases mais longas numa conversa banal e pedir ao usuário para explicar seus sentimentos sobre elas — essas frases tendem a ser importantes numa sentença simples. Os resultados melhoram quando se escolhem as palavras posteriores a "eu", "meu", "você" e "seu", por exemplo.

Escolher entre tais métodos e aperfeiçoá-los constitui fascinante exercício de programação. Você terá uma visão muito clara da imensa complexidade da linguagem humana e de sua análise e pode ficar insatisfeito com o BASIC como ferramenta de análise. Mas perceberá os aspectos positivos de linguagens de programação como LISP e PROLOG, estruturadas muito mais engenhosamente, sobretudo pela necessidade de lidar com as complexidades da linguagem — expressão do pensamento.

Além disso, você vai ter a oportunidade de conversar, pelo tempo que desejar, com o interlocutor perfeito — alguém que ouve.

GRAFIX 100MX

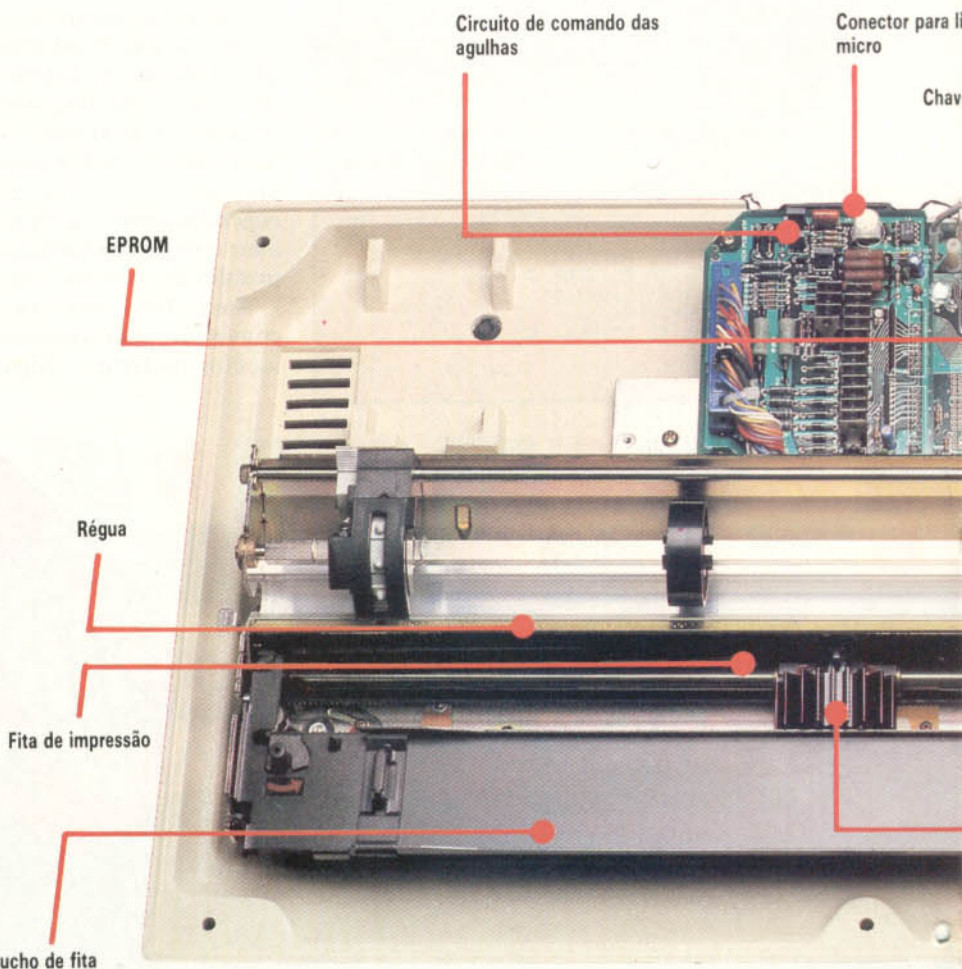
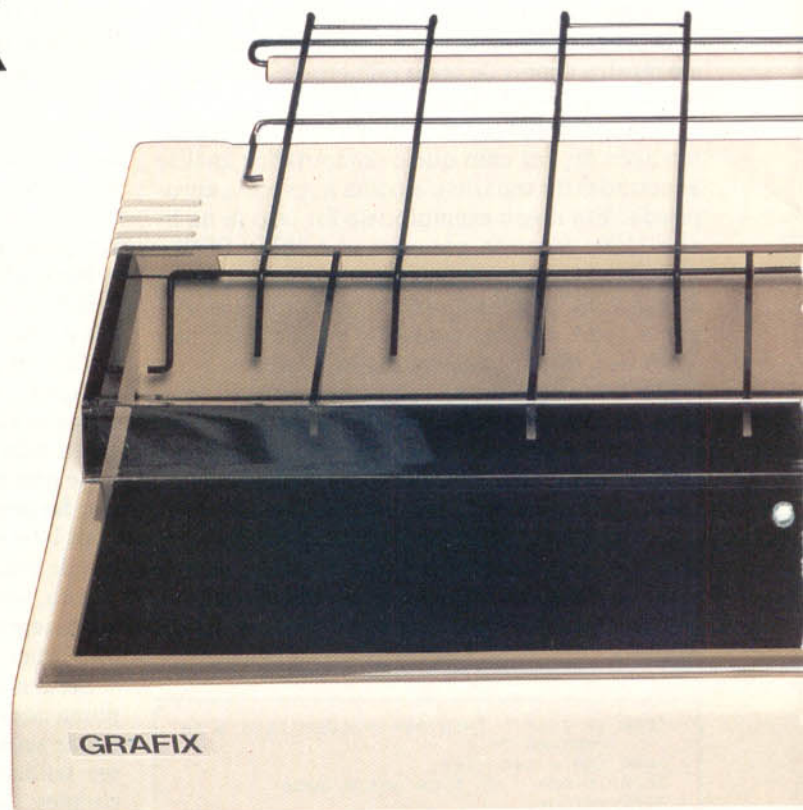
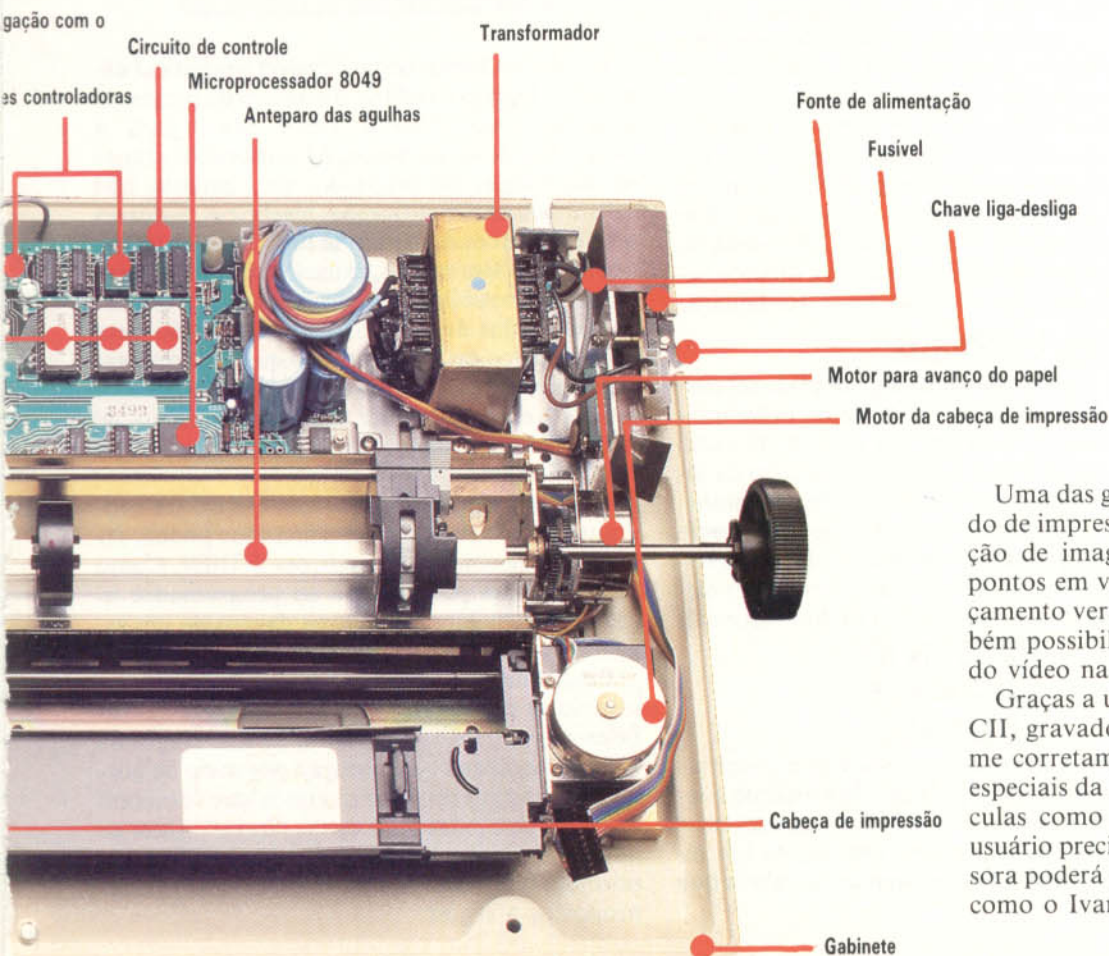
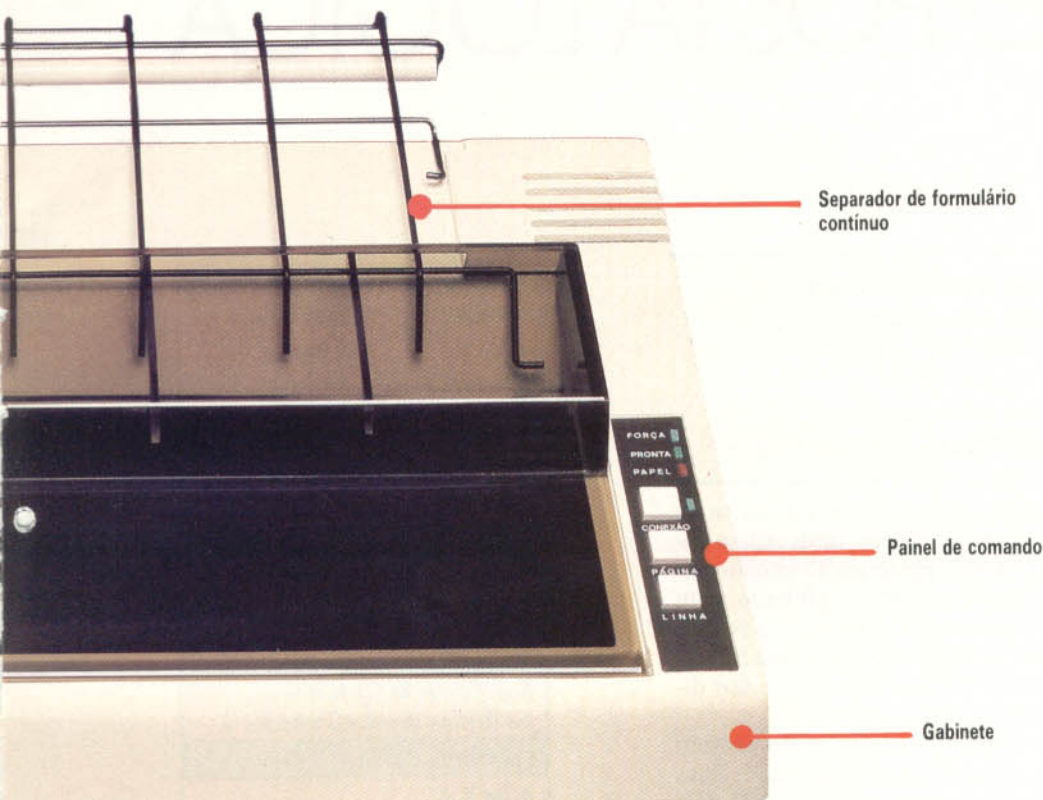
Amplos recursos para formatação de textos e variação de caracteres — além de uma grande resistência — são os maiores atrativos das impressoras matriciais Grafix MX para o usuário especializado.

O usuário que precisa de uma impressora com recursos para listagem de programas, processamento de textos e produção de gráficos encontra nos modelos da linha Grafix MX, fabricados pela Scritta, uma boa opção entre as alternativas disponíveis no mercado nacional.

As impressoras matriciais Grafix 80MX e 100MX possuem praticamente as mesmas características, distinguindo-se apenas pela velocidade de impressão — 80 cps (caracteres por segundo) na primeira e 100 cps na segunda — e pela largura máxima dos formulários contínuos com que trabalham: de 4 a 10 polegadas e de 4 a 15 1/2 polegadas. Equipadas com a interface paralela Centronics ou, opcionalmente, com a RS232C ou a IEEE488, podem ser acopladas à maioria dos micros.

Utilizando matriz de 9 x 9 agulhas e imprimindo em ambas as direções (com dispositivo de pesquisa lógica), os dois modelos são bastante versáteis quanto aos recursos para apresentação do texto, possibilitando tarefas especializadas, tais como tabelas financeiras com muitas colunas, títulos em destaque etc. Para tanto, além do modo de impressão normal, com 10 cpp (caracteres por polegada), o usuário pode escolher entre os modos comprimido (17,16 cpp) e expandido (5 cpp, em que a largura dos caracteres é duplicada), ou, ainda, caracteres com largura dupla comprimida (8,5 cpp).

Quando se requer impressão com ''qualidade de carta'' — ou seja, semelhante à das máquinas de escrever —, a Grafix pode imprimir em negrito, rebatendo cada caractere com um deslocamento de 1/126 de polegada, fazendo desaparecer os espaços entre os pontos da matriz de agulhas; ou no modo de impressão enfatizada, que também pode ser combinado com o negrito, embora nesse caso a velocidade reduza-se à metade da normal.

Outros recursos incluem texto em itálico (em todas as opções de tamanho de caractere), sobrescrito e subscrito (uma letra ou qualquer outro caractere menor acima ou abaixo da linha), sublinhado; tabulação horizontal e vertical; entrelinhamento variável etc. Essas variações da escrita são ativadas, temporária ou permanentemente, por comandos introduzidos pelo software do micro.

GRAFIX 100MX

DIMENSÕES
592 x 393 x 133 mm.

PESO
10 kg.

MÉTODO DE IMPRESSÃO
Impacto por matriz de pontos.

VELOCIDADE DE IMPRESSÃO
100 cps.

DIREÇÃO DE IMPRESSÃO
Bidirecional, com pesquisa lógica.

MODOS DE IMPRESSÃO
Normal, duplo (negrito), enfatizado e duplo enfatizado.

AGULHAS DA CABEÇA
9.

MATRIZ
9 x 9.

TIPO DE PAPEL
Formulário contínuo.

LARGURA DO PAPEL
4 a 15 1/2".

NÚMERO DE CÓPIAS
Um original mais duas cópias com carbono.

COLUNAS
Normal, 136; dupla largura, 68; comprimida, 233; dupla largura comprimida, 116.

CONJUNTO DE CARACTERES
255 ASCII.

INTERFACE
Paralela, tipo Centronics.

DOCUMENTAÇÃO
Manual do usuário, didático e minucioso.

Uma das grandes vantagens da Grafix é o modo de impressão gráfica, que permite a elaboração de imagens de alta resolução, utilizando pontos em vez de caracteres e variando o espaçamento vertical e horizontal. Além disso, também possibilita o despejo (dump) do conteúdo do vídeo na impressora.

Graças a um repertório de 255 caracteres ASCII, gravado em três EPROM, a Grafix imprime corretamente todos os acentos e caracteres especiais da língua portuguesa, tanto em maiúsculas como em minúsculas. No entanto, se o usuário precisar de caracteres especiais, a impressora poderá vir equipada com outros repertórios como o Ivanita, o ABICOMP e o Europeu.

A RESPOSTA LÓGICA

Antes de passar a teorias mais adiantadas, veja um resumo das regras da álgebra booleana e uma série de exercícios para revisão dos princípios lógicos nela contidos.

Nos computadores, os princípios da lógica se aplicam ao projeto do hardware, destinado a desempenhar certas tarefas especiais. O circuito somador, por exemplo, ao ser combinado com outros circuitos somadores, possibilita a adição de números binários, imitando o método humano de adição. Isso permite o transporte dos dígitos de uma coluna para a seguinte.

Utilizamos três elementos básicos no projeto desse circuito, chamados "portas lógicas". Suas funções foram indicadas por E, OU e NÃO. Cada uma delas se definiu graças a uma tabela de validação — meio simples de escrever a(s) saída(s) de um circuito para qualquer combinação possível de entradas. Duas entradas nos deram quatro (2^2) combinações possíveis de entradas, três entradas nos deram oito (2^3) combinações e assim por diante.

Também já vimos como as portas lógicas são ligadas para a obtenção das saídas desejadas. Esses circuitos mais complexos poderiam também ser descritos por suas tabelas de validação. Em especial, projetamos um circuito OU-exclusivo, que resultava numa saída verdadeira quando somente *uma* de suas entradas fosse verdadeira.

A álgebra booleana

A combinação de elementos lógicos pode ser descrita no papel por um conjunto de símbolos muito semelhantes aos da álgebra normal. O ramo da matemática vinculado à representação da lógica é conhecido como álgebra booleana, homenagem ao matemático inglês George Boole (1815-1864), que pela primeira vez definiu e sistematizou seus princípios. Cada um desses elementos básicos tem seu próprio símbolo especial:

A E B	$A.B$
A OU B	$A+B$
NÃO A	$\bar{A}$

Assim como a álgebra comporta regras próprias para controlar a manipulação de números e letras, também existem leis especiais que regem a simplificação das expressões lógicas. As leis da álgebra booleana estão resumidas na tabela que se segue:

RELAÇÕES ESPECIAIS	
Relação	**Dual**
$A.A=A$	$A+A=A$
$A.\bar{A}=0$	$A+\bar{A}=1$
$A.0=0$	$A+1=1$
$A.1=A$	$A+0=A$
$A.(A+B)=A$	$A+A.B=A$
$A.(\bar{A}+B)=A.B$	$A+\bar{A}.B=A+B$

LEIS DE DE MORGAN
1) $\overline{A+B}=\bar{A}.\bar{B}$
2) $\overline{A.B}=\bar{A}+\bar{B}$

LEI ASSOCIATIVA
$A.(B.C)=(A.B).C=A.B.C$
$A+(B+C)=(A+B)+C=A+B+C$

LEI COMUTATIVA
$A.B=B.A$
$A+B=B+A$

LEI DISTRIBUTIVA
$A.(B+C)=A.B+A.C$

O uso dessas regras torna possível simplificar expressões lógicas e reduzir o número de portas requeridas pelo circuito final. Mas, para a simplificação de circuitos, há também os mapas de Karnaugh, ou mapas-k, que, embora não substituam as simplificações algébricas, reduzem o esforço necessário para se lidar com a álgebra booleana. Esses mapas, extensões dos diagramas de Venn, possibilitam o agrupamento de expressões extraídas de uma tabela de validação para dois, quatros ou oitos (grupos que representam expressões booleanas mais simples). Na prática, muitas vezes se requer uma combinação de mapas-k com a álgebra para projetar o circuito mais eficiente.

Quanto às operações lógicas E e OU na programação em BASIC, esses comandos podem ser usados para combinar condições IF-THEN. Como já vimos, eles possibilitam ao programador ligar e desligar bits individuais dentro de um registro.

Finalizando o primeiro estágio do curso, sugeriu-se aplicar todo o conhecimento das tabelas de validação, dos mapas de Karnaugh, da álgebra booleana e da notação por meio de portas lógicas para projetar circuitos que fornecem as saídas desejadas para certas tarefas. Antes de lidar com teorias lógicas mais avançadas, é importante que você faça os exercícios de revisão propostos a seguir:

Exercício de revisão

1) Um conjunto de rock acredita no sucesso de sua mais recente gravação, desde que faça um bom videoclip *e* sua gravadora inclua um brinde no disco, *ou* desde que apareça num programa popular de tevê. Desenhe uma tabela de validação para todos os resultados possíveis. Se cada evento for igualmente provável, quais as possibilidades de o grupo vir a ter êxito?

2) O clube de xadrez de uma escola está admitindo novos sócios entre:

i) membros do corpo de funcionários;
ii) alunos do quarto *ou* quinto anos, que estudem matemática *e* ciências;
iii) alunos do sexto ano que estejam estudando matemática *ou* ciências.

Desenhe um emissor automático de carteiras de sócio que é operado pressionando-se os botões que melhor descrevem o candidato. Os botões são:

A = aluno do quarto ano;
B = aluno do quinto ano;
C = aluno do sexto ano;
D = membro do corpo de funcionários;
E = aluno estudante de matemática;
F = aluno estudante de ciências.

Faça um diagrama de circuitos para a máquina.

3) Certo microcomputador possui um registro com o endereço 23148 (decimal), utilizado para controlar a imagem no vídeo. O bit 0 é o menos significativo no registro e o bit 7 é o mais significativo. A tela pode ser passada para o modo de alta resolução colocando-se os bits 4 e 5 em um. Como os demais bits no registro são usados para controlar outras funções, é importante que seus valores não se alterem. Escreva comandos em BASIC que:

a) liguem a tela de alta resolução;
b) desliguem novamente a tela.

4) A caixa-forte de um banco funciona com um sistema de interruptores operados a chave. Três pessoas têm chave: o gerente, seu auxiliar e o tesoureiro. A porta se abre com duas das três chaves, mas:

a) entre 9 e 17h, é indispensável a chave do tesoureiro;
b) nos outros horários, servem a do gerente e qualquer das duas restantes.

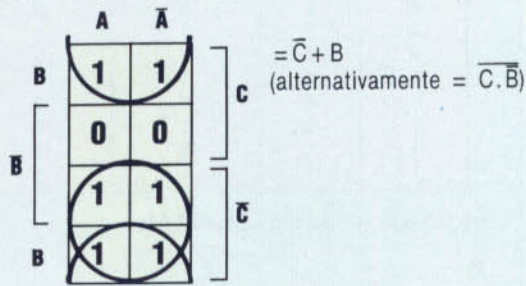
Desenhe um circuito para controlar o sistema.

5) Numa máquina de votar, cada bit de dados digitais binários é enviado simultaneamente para três canais paralelos que, na extremidade receptora, são fornecidos a um painel eletrônico. Não ocorrendo erros de transmissão, os três bits deverão ser idênticos. Caso contrário, o receptor de votos corrigirá o erro num canal dando uma saída simples, correspondente à maioria das entradas. Desenhe um circuito para o receptor.

Respostas do exercício 5

1a)

	A	$\bar{A}$	
B	1	1	C
$\bar{B}$	0	0	C
$\bar{B}$	1	1	$\bar{C}$
B	1	1	$\bar{C}$

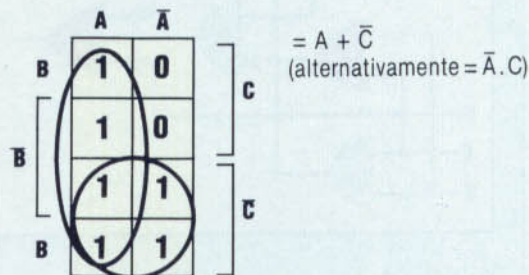
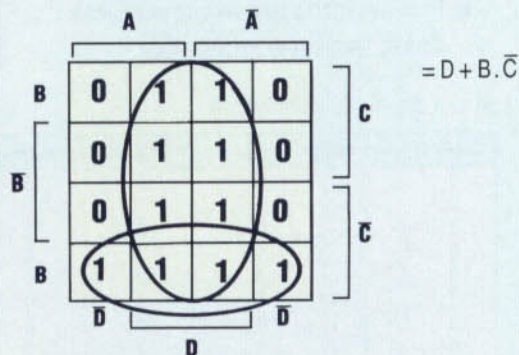
$=\bar{C}+B$
(alternativamente $=\overline{C.\bar{B}}$)

b)

$\overline{B+C}+B.\bar{C}+A.C$
$=\bar{B}.\bar{C}+B.\bar{C}+A\,C$ (De Morgan)

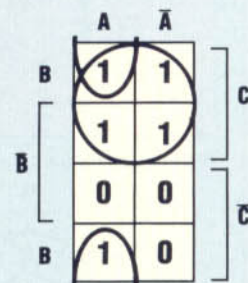
	A	$\bar{A}$	
B	1	0	C
$\bar{B}$	1	0	C
$\bar{B}$	1	1	$\bar{C}$
B	1	1	$\bar{C}$

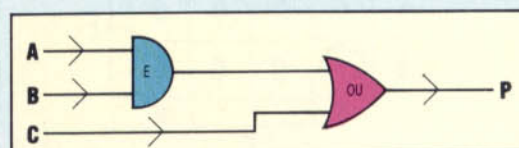
$= A + \bar{C}$
(alternativamente $=\overline{\bar{A}.C}$)

c)

	$\bar{D}$ (A)	D (A)	D ($\bar{A}$)	$\bar{D}$ ($\bar{A}$)	
B	0	1	1	0	C
$\bar{B}$	0	1	1	0	C
$\bar{B}$	0	1	1	0	$\bar{C}$
B	1	1	1	1	$\bar{C}$

$=D+B.\bar{C}$

2) A tabela de validação é:

Decimal	A	B	C	P
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

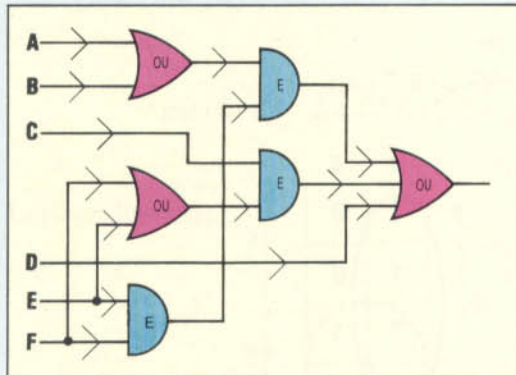
	A	$\bar{A}$	
B	1	1	C
$\bar{B}$	1	1	C
$\bar{B}$	0	0	$\bar{C}$
B	1	0	$\bar{C}$

Do mapa-k obtém-se a expressão $P=C+A.B$. O circuito resultante é:

A B C E OU P

Resposta do exercício de revisão

1)

Bom vídeo	Brinde	Show na tevê	Êxito de vendas
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Probabilidade de êxito = 5/8 = 62,5%

2)

3 a) Para ligar a tela de alta resolução:

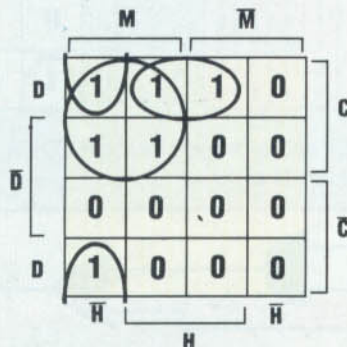
POKE23148,PEEK(23148)OU48

b) Para desligar a tela de alta resolução:

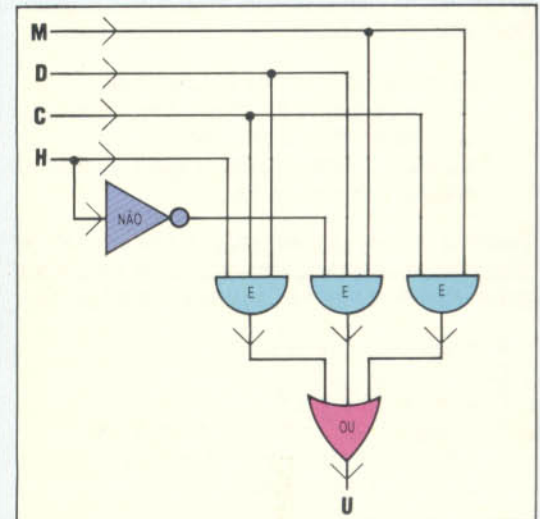
POKE23148,PEEK(23148)OU207

4) A tabela de validação é:

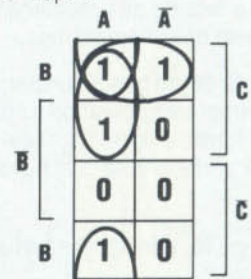
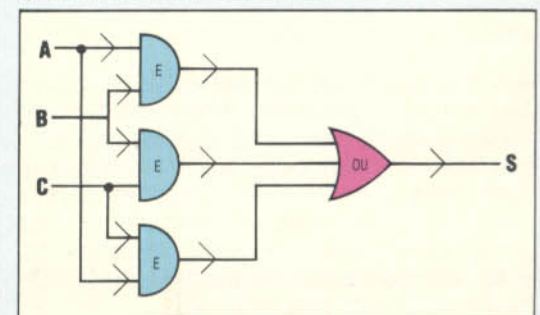
Gerente (M)	Auxiliar (D)	Tesoureiro (C)	Das 9 às 17h (H)	Abre a porta (U)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

A partir do mapa-k

obtemos a expressão: $U = M.C + M.D.\overline{H} + D.C.H$.
Para nos certificarmos de que não foi cometido algum erro, o melhor é reconverter essa expressão para frases em português. Segundo a expressão, a porta pode ser aberta por:

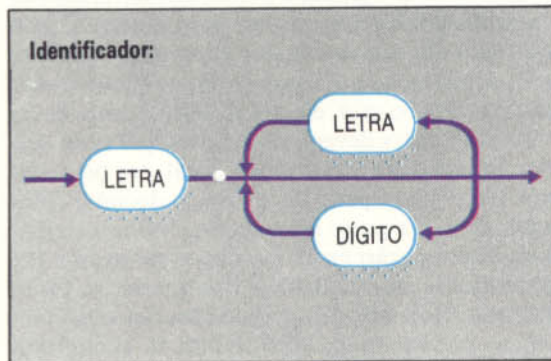
i) o gerente e o tesoureiro, a qualquer hora;
ii) o gerente e o auxiliar, fora do expediente; e
iii) o auxiliar e o tesoureiro, das 9 às 17h.

Eis o circuito correspondente:

5) A tabela de validação é:

A	B	C	S
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	[illegible]
1	1	1	1

Com a ajuda do mapa-k

obtemos a expressão $S = A.B + A.C + B.C$, que corresponde a este circuito:

DADOS E DECISÕES

Quatro tipos de dados, instruções compostas e estruturas de tomada de decisões são os recursos de programação que examinaremos neste artigo da série sobre a poderosa linguagem PASCAL.

O PASCAL dispõe de quatro tipos simples de dados — ou seja, que podem ter apenas um valor —, representados pelos identificadores INTEGER, REAL, CHAR e BOOLEAN. Os números são classificados em fracionários (números reais) ou inteiros naturais (em inglês, integer). A faixa efetiva dos números disponíveis é limitada pela quantidade de bytes usada para armazenar determinado valor de cada um desses tipos. Cabe ao projetista do compilador (o "implementador") definir essas características, chamadas "dependentes da implementação". Tudo o que depende da implementação deve vir especificado na documentação segundo o padrão ISO (International Standards Organization) do PASCAL.

A faixa dos integers em seu compilador provavelmente se estende de −32.768 a 32.767 ou de −2.147.483.648 a 2.147.483.647, conforme seja usada representação de 2 ou de 4 bytes. No PASCAL representa-se o valor do número integer máximo pelo identificador de constante predefinido, MAXINT. Esse valor pode ser facilmente encontrado por meio da instrução:

```
WRITELN ('O MAXIMO INTEGER E:',MAXINT)
```

Os números reais também se restringirão a uma faixa e exatidão determinadas — em geral, de cerca de 1,7E+38 —, com seis ou sete dígitos de precisão, no mínimo. Essa forma de escrever os números reais, conhecida como "notação científica", é o padrão preestabelecido do PASCAL. Mas você pode usar o método mais comum de escrevê-los com uma vírgula decimal (por exemplo, 123,456).

CHAR é a abreviação de character (caractere), e um valor desse tipo será um dos elementos do conjunto de caracteres (geralmente ASCII) utilizado pelo computador. O PASCAL garante sua própria portabilidade ao supor que:

- Os caracteres de A a Z estarão ordenados alfabeticamente; isso significa que, em termos de valor de caractere, A será menor que B, este menor que C etc.
- Os caracteres numéricos de 0 a 9 estarão ordenados e contíguos; isto é, qualquer que seja o valor de 0, o 1 virá em seguida, e assim sucessivamente.

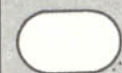
Cada valor de caractere terá um código numérico correspondente a um valor associado a uma subfaixa do tipo integer. Os códigos ASCII são definidos na faixa de 0 a 127, embora em muitas máquinas com códigos adicionais para caracteres gráficos esse limite seja ampliado para 225. Podemos estabelecer uma correspondência entre qualquer conjunto ordenado de caracteres e a escala de valores "ordinais" usada internamente pelo computador. Para isso, o PASCAL utiliza a função predefinida ORD: ela fornece o código integer de seu argumento — assim, ORD (A) equivale a 65 no conjunto de caracteres ASCII. Uma outra função, CHR, fornece o mapeamento no sentido inverso — CHR (65) gera o caractere A (note que o cifrão, $, *não* é usado com CHR).

A faixa de ambos — valores de caracteres e integers — é específica para cada equipamento e corresponde a uma escala ordenada de constantes conhecidas. Por esse motivo, são eles denominados "tipos ordinais" ou "escalares". Qualquer que seja o valor, sempre saberemos os valores seguintes, se existirem. Obtêm-se esses valores por meio de duas funções escalares:

```
PRED (ITEM) (PREDECESSOR)
SUCC (ITEM) (SUCESSOR)
```

Assim, SUCC (3) corresponderá ao valor de caractere 4, mas PRED (Z) corresponderá apenas a Y em certos conjuntos de caracteres, como no ASCII, por exemplo. PRED (MAXINT) será 32.766 ou 2.147.483.646. Emprega-se a função CHR apenas com argumentos que sejam códigos de caracteres. Todas as outras funções escalares aceitam qualquer tipo escalar, embora ORD, ao ser usada com integers, forneça apenas o valor de seu argumento.

As variáveis booleanas são os tipos escalares mais simples de todos, pois há apenas dois valores na escala — falso (FALSE) e verdadeiro (TRUE). Assim, as funções escalares aplicam-se

Símbolos do PASCAL
Estas três formas são os símbolos mais usados nos diagramas de sintaxe:

- A de cantos arredondados ou em forma de losango representa as palavras reservadas, ou caracteres que não necessitam de outras explicações.
- Um círculo indica um operador do PASCAL (+,−,.,* etc.).
- O retângulo indica uma palavra ou frase que possui seu diagrama de sintaxe separado.

a qualquer valor booleano — 0 é o valor ordinal de FALSE e 1 o de ORD (TRUE). As outras funções escalares, PRED e SUCC, não são muito úteis por enquanto.

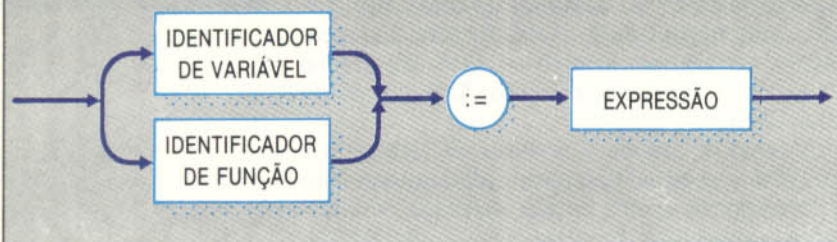
O exemplo que segue de definição de constante de um programa mostra todos os tipos ordinais simples tal como aparecem no texto-fonte do PASCAL.

```
CONST
  VAT         =0,15;
  COLUNAS     =40;
  ESPACO      =' ';
  DEBUGGING   =FALSE;
```

Igualdade e atribuição

O símbolo de igualdade (=) sempre significa "igual" em PASCAL e é usado para equiparar os identificadores de constante com seus valores. Quando introduzimos variáveis na seção de declaração VAR, o sinal de dois-pontos (:) separa o identificador de variável de seu tipo. Por exemplo:

```
VAR
  TAXA        : REAL;
  NUMERO      : INTEGER;
  SIMBOLO     : CHAR;
  FEITO       : BOOLEAN;
```

Para atribuir valores a essas variáveis, usa-se o "operador de atribuição" (:=) composto.

Isso ajuda a distinguir com clareza entre as três classes de operação. As definições CONST equalizam valores permanentes, as declarações VAR somente reservam espaço na memória, e ATRIBUICAO fornece um valor (possivelmente temporário) ao identificador.

Instruções compostas

Quando se executam duas ou mais instruções como parte de um único processo, elas são colocadas entre as palavras BEGIN e END, constituindo uma instrução "composta". Lembre-se de que cada instrução deve ser separada da seguinte por meio de ponto-e-vírgula. Já vimos uma instrução composta, uma vez que o núcleo de instruções executáveis de qualquer programa assume essa forma. Eis um programa completo que utiliza muitos dos recursos citados:

```
PROGRAM CIRCULO (INPUT, OUTPUT);
CONST
  PI          =3.1415926536;
  QUESTAO     ='ENTRE COM O RAIO:';
VAR
  RAIO,
  AREA        : REAL;
BEGIN
  WRITELN;
  WRITE (QUESTAO);
  READ (RAIO);
  AREA: = PI * RAIO * RAIO;
  WRITELN;
  WRITELN ('A AREA DO CIRCULO',
    'DE RAIO';RAIO : 8 : 3);
  WRITELN ('E :', AREA : 10 : 3);
END.
```

Em primeiro lugar, a parte VAR apresenta dois identificadores do mesmo tipo — ambos reais. Não são, no entanto, necessárias duas declarações separadas, pois uma vírgula separa as listas de itens. Do mesmo modo, quando especificamos mais de um argumento num procedimento — como nas instruções WRITELN —, a mesma regra se aplica. A segunda característica inovadora está na formatação da saída, que dá prioridade à notação científica dos valores reais. Temos a opção de especificar duas variáveis de integers separadas por sinais de dois-pontos, a fim de impor certo tamanho de campo para o número inteiro e sua parte fracionária.

Em nosso programa Círculo, atribuiremos três casas decimais para o raio e para a área. Como esta será um número maior (portanto mais longo), deixaremos um total de dez posições de caracteres para o raio, em vez de apenas oito. Esses valores integers devem ser maiores que zero, deixar espaço para um possível sinal, ter pelo menos um dígito e permitir a vírgula decimal colocada antes da parte fracionária. Valores ilegítimos ocasionarão erros quando o programa for processado, ou, na melhor das hipóteses, farão voltar à padronização original em notação científica; por exemplo, WRITELN (X : 6 : 2) não deixaria espaço para números maiores que 99,99.

Outro recurso ainda mais útil do PASCAL é o arredondamento automático do último dígito, possibilitando o máximo de precisão em qualquer campo numérico requisitado. Além disso, pode-se usar qualquer variável ou expressão, e não apenas uma constante. Isso permite enorme flexibilidade, inclusive recursos de tabulação. Com todos os demais tipos de dados, somente um valor integer é necessário — ou melhor, permitido — para especificar o tamanho do campo. Tal especificação fará com que os valores integers sejam escritos no campo de tamanho mínimo, sem espaços. Assim, teremos de nos lembrar de colocá-los nós mesmos, se quisermos a tabulação dos resultados. Por exemplo:

```
WRITELN ('TOTAL : ' : 20, PESO : 1, 'TONS.')
```

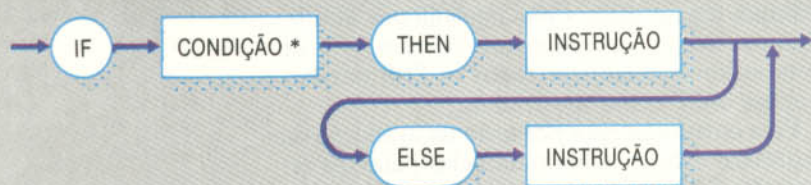
O PASCAL normalmente se recusará a fornecer um resultado impreciso, mas a supressão de to-

dos os espaços exige muito cuidado para que não sejam impressos dois números consecutivos num campo para um único número. Por exemplo, se 12 e 34 forem escritos dessa forma, o resultado será 1234.

Estruturas de decisão

A instrução IF do PASCAL é usada de modo semelhante à maioria das linguagens. Como o final de uma linha não encerra uma instrução, podemos usar a formatação livre do PASCAL para demonstrar as possíveis rotas através da estrutura da instrução IF. Eis duas dessas instruções:

```
IF CONTADOR = LIMITE
  THEN
    WRITELN ('SEM ESPACO')
  ELSE
    WRITE ('PROXIMO?');
IF NUMERO > MAXIMO THEN
  MAXIMO := NUMERO



ELSE
  {NENHUMA ACAO}
```

O alinhamento das palavras reservadas THEN e ELSE (quando presentes) facilita a visualização do fluxo de controle pela "estrutura". O ponto-e-vírgula é usado para separar instruções, como mostra o exemplo; portanto, *nunca* pode ser colocado antes de uma instrução ELSE. Lembre-se de que ELSE não é um comando em PASCAL, mas apenas IF; THEN e ELSE são usadas para delimitar a condição booleana e as instruções nas duas sentenças. Toda vez que usamos uma instrução IF, testamos um valor booleano. A "sentença" THEN só será executada se a expressão for verdadeira. Caso ela seja seguida por uma sentença ELSE, a instrução (ou instruções) contida nessa sentença será executada quando o valor booleano for falso.

O PASCAL foi a primeira linguagem a introduzir os dados conceituais conhecidos como "escalares numerados". Eles são úteis porque possibilitam uma visão abrangente da classificação dos dados, sem que precisemos traduzi-los mentalmente para códigos numéricos. Já vimos como, nesse caso, a simples definição de constante pode ser útil:

```
CONST
  LARGURA = 80;
```

O identificador de constante LARGURA pode então ser usado em todo o programa para referência ao número de colunas no sentido da largura da tela ou da impressora. Refazer o programa para um vídeo de 40 colunas torna-se assim simples questão de modificar a definição no início do programa. Uma escala completa de constantes define-se por meio desse método.

Se estivéssemos usando gráficos coloridos, as cores disponíveis seriam associadas a uma escala de números (vermelho = 1, verde = 2, por exemplo), o que — teoricamente — permitiria a extração da raiz quadrada de verde, ou a multiplicação de azul por amarelo. Isso, além de absurdo, é também uma fonte de erros em potencial quando estamos resolvendo problemas que não envolvem apenas dados numéricos. A parte da definição TYPE de um programa em PASCAL pode ser usada para definir um tipo escalar conceitual inteiramente novo, pela simples enumeração de uma lista de identificadores representando os valores constantes da escala. Por exemplo:

```
TYPE
  MATIZ = (VERMELHO, VERDE, AMARELO,
    AZUL);
```

Como se trata de um novo tipo de definição e não de uma declaração, a sintaxe utiliza o sinal de igualdade para definir o identificador de tipo (MATIZ) e indicar os valores ordenados do tipo entre parênteses. De modo análogo, o tipo predeterminado integer refere-se a todos os números integers existentes na implementação do PASCAL.

Como sempre acontece no PASCAL, os identificadores sucessivos numa lista (nesse caso, valores para cores) apresentam-se separados por vírgulas. Esses valores conceituais são automaticamente associados (pelo compilador) a valores integers, com a enumeração ordinal partindo de 0. Assim como o caractere A equivale a 65 no código ASCII, cada valor de MATIZ corresponde a um número ordinal, obtido por meio da função escalar ORD. No exemplo, ORD (VERMELHO) é 0, e ORD (AZUL) equivale a 3. Definido esse novo tipo escalar, declaramos uma variável de forma habitual:

```
VAR
  COR : MATIZ;
```

Essa estrutura declara que um identificador (COR) constitui um item de dados do tipo MATIZ, da mesma forma que a declaração

```
VAR
  LETRA : CHAR;
```

associa o tipo caractere ao objeto de dados chamado LETRA. As únicas operações definidas nos tipos enumerados são os testes de relação e o uso das funções escalares. Por exemplo, poderíamos escrever:

```
IF COR < AZUL THEN
  COR : = SUCC (COR)
```

Lembre-se de que PRED (VERMELHO) e SUCC (AZUL) não existem. A variável COR é incompatível com variáveis de qualquer outro tipo, escalares ou não. Isso significa que não podemos executar funções ilegítimas como extrair a raiz quadrada ou dizer:

```
COR: = COR + 1
```

Disso resulta uma limitação óbvia. Os caracteres e os números podem ser usados como parâmetros para instruções WRITE e WRITELN, mas

```
WRITELN (COR)
```

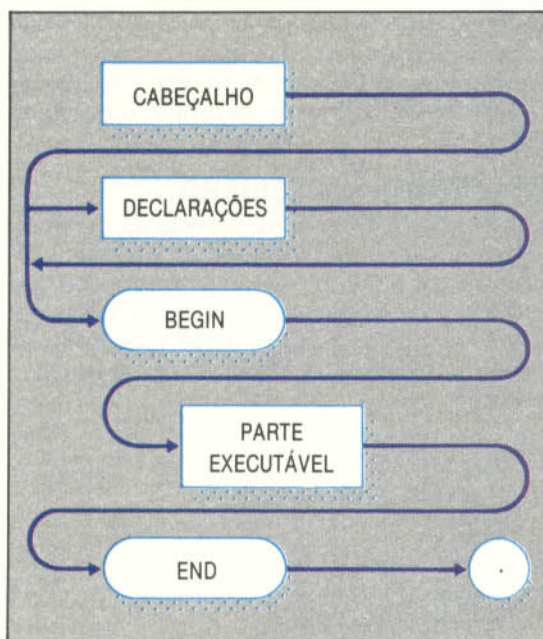
seria ilegítima, porque os valores do tipo são puramente conceituais e, se quisermos imprimir seus nomes, teremos de associar os valores de cores às cadeias de caracteres. Essa é uma utilização ideal para outra estrutura especial do PASCAL, a instrução CASE.

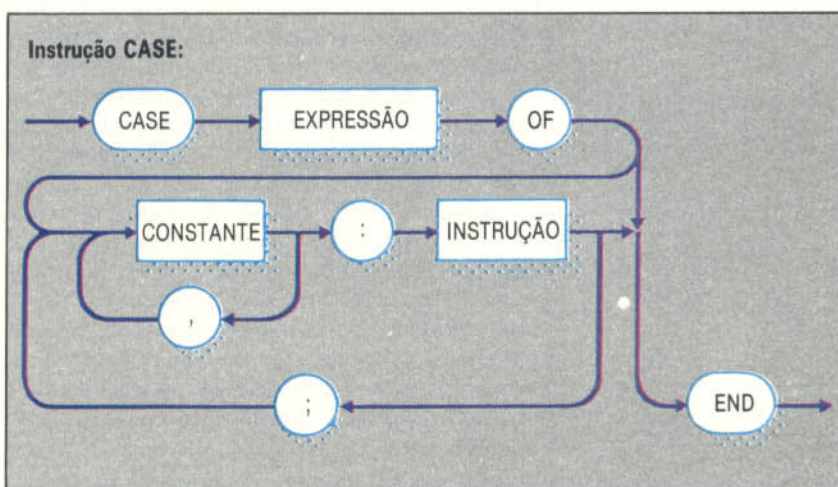
Vimos como, em PASCAL, a instrução IF parece bastante familiar e é facilmente visualizada graças às convenções de formatação livre. Contudo, há ocasiões em que devem ser tomadas decisões de opções múltiplas, que exigiriam algo do tipo:

```
IF N = 1
  THEN
    WRITE ('PRIMEIRO')
  ELSE
    IF N = 2
      THEN
        WRITE ('SEGUNDO')
      ELSE
        IF N = 3
          THEN
            WRITE ('TERCEIRO')
          ELSE
            WRITE ('QUARTO')
```

Sempre que a instrução a ser executada depender de o valor de um escalar simples estar num certo limite restrito, obteremos melhores resultados se empregarmos a instrução CASE. Nesse caso:

```
CASE N OF
  1        : WRITE ('PRIMEIRO');
  2        : WRITE ('SEGUNDO');
  3        : WRITE ('TERCEIRO');
  4,5,6
  7,8,9    : WRITE ('QUARTO')
END CASE
```

Note que isso será conveniente apenas para os valores de N na faixa dos "rótulos case" especificados no "núcleo" da instrução CASE — 1 a 9, no exemplo. Todos os valores que N poderá ter quando o programa for executado *deverão* estar relacionados individualmente, e não seria aceitável, por exemplo, introduzir N = 0.

Esquema do programa
Este é o diagrama básico da sintaxe de um programa em linguagem PASCAL, apresentando todos os seus elementos importantes. Os programas em PASCAL podem ser escritos com letras maiúsculas ou minúsculas; em algumas versões, apenas as palavras reservadas vêm em maiúsculas.

Muitos compiladores PASCAL também possuem a palavra reservada OTHERWISE ou OTHERS, que serve para relacionar um procedimento previamente padronizado.

Essa é a única instrução PASCAL onde END delimita uma estrutura não iniciada por BEGIN. Trata-se de um procedimento comum — e de bom estilo — qualificar todos os END das instruções CASE como mostramos.

Essa estrutura opera por meio da avaliação da expressão escalar delimitada pelas palavras reservadas CASE e OF. Um nome de variável simples é uma expressão mínima que não exige cálculos. O valor obtido é então comparado com cada constante relacionada nas listas de rótulo CASE e, ao ser encontrada uma correspondência, a instrução que vem após os dois-pontos — e somente essa instrução — é executada. Assim, o fluxo de controle não se "desvia", preservando a integridade estrutural. Se for necessária a execução de várias operações, emprega-se uma instrução composta, colocada entre o par BEGIN/END.

Em certas circunstâncias, alguns valores não requerem a execução de uma atividade, caso em que se pode usar a mais simples de todas as instruções do PASCAL a NULL, que significa "nada executar" e não envolve absolutamente sintaxe alguma. Eis um exemplo:

```
CASE N MOD 4 OF
  0 : {NADA EXECUTAR};
  1,3 : BEGIN
    WRITE (N MOD 4 : 1, 'QUARTO');
    IF N MOD 4>1 THEN
      WRITE ('S')
    END;
  2 : WRITE ('METADE')
END {CASE}
```

O ponto-e-vírgula ainda é necessário para separar essa instrução não existente daquela que vem a seguir, com exceção da última, pois ela vem antes de uma palavra reservada (END) e não de outro rótulo ou instrução. O uso do operador MOD na expressão garante que o valor esteja dentro do limite de 0 a 3. Ele fornece o resto de uma divisão integer, como acontece em várias versões do BASIC.

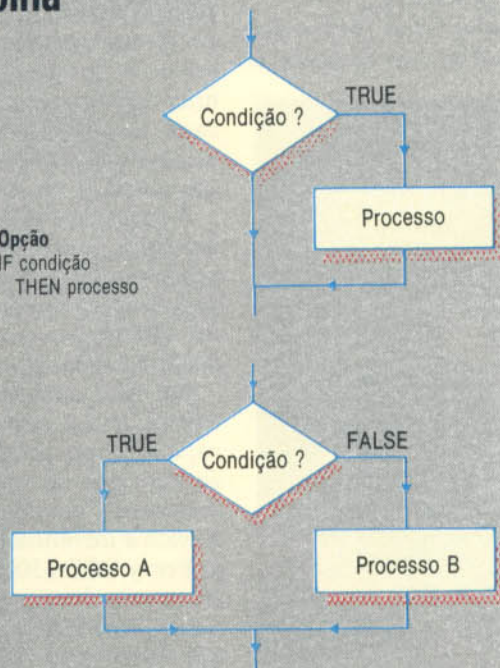
No próximo artigo examinaremos esse e os demais operadores do PASCAL, bem como as funções incorporadas. Por enquanto, aqui está a solução para o problema de como imprimir as cadeias de caracteres para cada valor de nosso próprio tipo de cor e matiz:

```
CASE COR OF
  VERMELHO  : WRITE ('VERMELHO');
  VERDE     : WRITE ('VERDE');
  AMARELO   : WRITE ('AMARELO');
  AZUL      : WRITE ('AZUL');
END {CASE}
```

Estrutura de escolha

A estrutura IF-THEN executará o "processo" se a condição for verdadeira. Caso seja falsa, o programa simplesmente se desviará para a instrução seguinte.

Enquanto a estrutura IF-THEN apresenta uma "opção", em IF-THEN-ELSE ocorre uma "escolha". Conforme o resultado do teste realizado no início da estrutura, um dos dois processos será executado.

Opção
IF condição
THEN processo

Escolha
IF condição
THEN processo A
ELSE processo B

Instruções para o mês

Este programa, que calcula o número de dias entre uma data fornecida pelo usuário e o final do mês, apresenta alguns dos principais recursos da programação em PASCAL.

A instrução CASE é usada para associar os dados numéricos de entrada aos tipos numerados do PASCAL, usando-os para o processamento e depois associando-os de novo à cadeia de dados de saída. Note o uso da variável CHAR para ler qualquer caractere que separe os números introduzidos.

```
PROGRAM    DATA    (INPUT, OUTPUT);

CONST
  PONTO FINAL = '.';

TYPE

  CALENDARIO = (JAN, FEV, MAR, ABR, MAI,
  JUN, JUL, AGO, SET, OUT, NOV, DEZ);

VAR
  NOMEMES         : CALENDARIO;
  DIA,
  MES,
  ANO,
  FALTAM          : INTEGER;
  SIMBOLO         : CHAR;
  BISSEXTO        : BOOLEAN;


BEGIN
  WRITELN ('ENTRE COM A DATA NA FORMA :');
  WRITELN ('DD/MM/AA' : 40);
  WRITELN;
  WRITE ('DATA = >');
  READ (DIA, SIMBOLO, MES, SIMBOLO, ANO);
  BISSEXTO := ANO MOD 4 = 0;
  IF (MES >0) AND (MES < = 12)
    THEN
      CASE MES OF

        1           : NOMEMES := JAN;
        2           : NOMEMES := FEV;
        3           : NOMEMES := MAR;
        4           : NOMEMES := ABR;
        5           : NOMEMES := MAI;
        6           : NOMEMES := JUN;
        7           : NOMEMES := JUL;
        8           : NOMEMES := AGO;
        9           : NOMEMES := SET;
        10          : NOMEMES := OUT;
        11          : NOMEMES := NOV;
        12          : NOMEMES := DEZ;

      END;          {CASE}

    ELSE
      BEGIN
        WRITELN   ('EPA!');
        WRITELN   ('O PROGRAMA ASSIM');
        WRITELN   ('NAO FUNCIONA!');
      END;


    CASE NOMEMES OF
    JAN, MAR,
    MAI, JUL        : FALTAM := 31-DIA;
    ABR, JUN,
    SET, NOV        : FALTAM := 30-DIA;
    FEV             : IF BISSEXTO
                        THEN
                          FALTAM := 29-DIA;
                        ELSE
                          FALTAM := 28-DIA;


  END; {CASE}
  WRITELN ('FALTAM', FALTAM : 1,
      'DIAS PARA TERMINAR');

      CASE NOMEMES OF
      JAN           : WRITE ('JANEIRO');
      FEV           : WRITE ('FEVEREIRO');
      MAR           : WRITE ('MARCO');
      ABR           : WRITE ('ABRIL');
      MAI           : WRITE ('MAIO');
      JUN           : WRITE ('JUNHO');
      JUL           : WRITE ('JULHO');
      AGO           : WRITE ('AGOSTO');
      SET           : WRITE ('SETEMBRO');
      OUT           : WRITE ('OUTUBRO');
      NOV           : WRITE ('NOVEMBRO');
      DEZ           : WRITE ('DEZEMBRO');


      END; {CASE}
      WRITELN (PONTO FINAL);
    END.
```

ELEGÂNCIA ITALIANA

No mercado competitivo dos equipamentos para escritório, uma multinacional com base européia, a Olivetti, conseguiu se impor como fabricante de máquinas elegantes e de qualidade.

O M20
O computador M20, máquina de 16 bits, foi lançado no mercado em 1981. Incorpora um microprocessador Zilog Z8000 e usa o sistema operacional PCOS da Olivetti.

Camillo Olivetti fundou sua empresa em 1908. Com vinte empregados, montou uma pequena fábrica em Ivrea, no norte da Itália, e deu início à produção de sua primeira máquina de escrever, a M1.

Embora nessa época a economia italiana fosse predominantemente agrícola, a Olivetti cresceu, passando de quatro máquinas produzidas por dia, em 1914, para cinqüenta, em 1929.

Na década de 30, Adriano Olivetti, filho de Camillo, começou a reestruturar a companhia, empregando alunos da escola noturna da própria Olivetti, fundada em 1924. Foram também construídas moradias para os empregados. A empresa preocupava-se com a promoção de benefícios sociais e aplicou uma política funcional de assistência ao trabalhador.

Enquanto as indústrias do resto do mundo lutavam para sobreviver — o período era de depressão econômica em todo o mundo —, a Olivetti continuava crescendo. Em 1933, havia vendido aproximadamente 15 milhões de produtos para escritório. Quatro anos depois, lançou com êxito sua primeira teleimpressora, seguida, em 1940, pela primeira calculadora.

A Segunda Guerra Mundial (1939-1945) trouxe uma estagnação temporária, mas no período posterior ao conflito a empresa concentrou-se no desenvolvimento de novos mercados internacionais, com um êxito que teve como fundamentos principais a qualidade dos produtos e seu design elegante.

Nas décadas de 50 e 60, a Olivetti voltou-se para os computadores. Esse processo começou em 1955, com uma máquina para cálculos. Seu primeiro computador de grande porte, o Elea, foi desenvolvido alguns anos depois.

A empresa continuou a diversificar, produzindo também aparelhos eletrônicos. Uma nova linha de minicomputadores foi lançada, juntamente com terminais bancários e equipamentos de comunicação.

Atualmente, a Olivetti fabrica grande variedade de equipamentos profissionais para escritório e investe somas consideráveis no desenvolvimento de software apropriado para suas máquinas. Em 1982, foi a segunda maior indústria de computadores da Europa (só ultrapassada pela IBM), vendendo bem tanto seu portátil M10 quanto o computador profissional M20.

Na ocasião, a empresa italiana já planejava lançar uma máquina compatível com o IBM PC, mas de custo inferior. Denominada M24, teria como características um processador 8086-2 e a opção de aceitar uma placa Z8001 para torná-la compatível com o M20 da própria Olivetti. Em função disso, porém, a indústria precisou abandonar seu sistema operacional PCOS.

Empreendimentos brasileiros

A Olivetti instalou-se no Brasil em 10 de novembro de 1952. Sete anos depois, inaugurou sua fábrica em Guarulhos (SP) e iniciou a comercialização de máquinas de escrever, de calcular (mecânicas e eletromecânicas) e de contabilizar, assim como as de telex.

A década de 60 caracterizou-se, para a Olivetti brasileira, pelos equipamentos eletrônicos. Os primeiros foram as máquinas contábeis e as centrais de telecomunicação, seguindo-se calculadoras eletrônicas programáveis, terminais para teleprocessamento e fotocopiadoras.

Em 1985, eram três os produtos eletrônicos da Olivetti que se destacavam no mercado brasileiro: a máquina de escrever ET 121, do tipo margarida, que se transforma numa impressora acoplável a um micro, com seletores e teclas de serviço e memória de uma linha; o Sistema Data Entry SDE-2500; e a teleimpressora TE 520. Na mesma época, a empresa planejava lançar a máquina de escrever eletrônica portátil Praxis 20, que vinha sendo fabricada exclusivamente para exportação.

Arquitetura compatível
A Olivetti sempre foi conhecida pelo avançado design de seus produtos, conceito que se estende também à arquitetura dos edifícios da empresa. As instalações da foto são da Olivetti brasileira.

XADREZ NOS CHIPS

Batalha no tabuleiro
Mestre internacional de xadrez, David Levy abandonou as competições contra jogadores humanos em 1978. Em 1968, apostou enorme quantia em sua habilidade, desafiando qualquer programa de xadrez para computador a vencê-lo nos dez anos seguintes. O período abrangido pela aposta ampliou-se, mas o enxadrista permanece invicto. Considerado uma autoridade em xadrez para computador, Levy dirige a Intelligent Software, empresa que pesquisa técnicas de programação interativas. Muitos dos computadores dedicados ao xadrez e programas para xadrez em micro baseiam-se nessas técnicas. Levy acredita que, em termos de desempenho no jogo de xadrez, os micros aproximam-se agora dos computadores de grande porte.

Esta matéria examina o raciocínio e algumas das idéias que possibilitaram o desenvolvimento de programas para um dos mais antigos e conceituados jogos de estratégia em tabuleiro.

Poucos jogos seduziram a imaginação tanto quanto o xadrez: milhões de pessoas em todo o mundo o jogam há milhares de anos e suas regras atuais são quase as mesmas usadas desde o século XVII. Há os que dedicam a vida ao estudo desse jogo de estratégia, encontrando satisfação em sua exigência de agilidade mental.

O xadrez originou uma série de variações que buscam introduzir níveis maiores de complexidade. Por exemplo: o xadrez tridimensional envolve vários tabuleiros suspensos no espaço e exige concentração muito maior. Outra variante é o xadrez com três pessoas, jogado num tabuleiro em forma de Y. Nas diagonais, onde as três "alas" se intersectam, há regras especiais para o movimento das peças. A teoria por trás dessa versão é de que dois jogadores se unam contra um terceiro e depois lutem um contra o outro pela vitória. Mas nenhuma dessas variações conseguiu substituir o confronto entre duas pessoas jogando no tabuleiro com 64 quadrados.

Uma das razões disso está no número quase infinito de variações possibilitadas pelo próprio jogo. Em 1949, o matemático Claude Shannon calculou que há 10^{120} jogos possíveis com quarenta movimentos cada um. Isso significa que uma pessoa que jogasse xadrez 24 horas por dia, sete dias por semana, despendendo uma hora para cada jogo (o que não é tempo demais para quarenta movimentos), levaria mais de 10^{17} anos para fazer os jogos possíveis.

Dada a complexidade, não é de surpreender que a programação de computadores para jogar xadrez tenha exigido muito tempo e esforço. Os programas de xadrez já são executados em computadores de grande porte há muitos anos e existem agora inúmeras versões para micros.

O desenvolvimento de programas de alta qualidade para micros vincula-se às inovações no hardware. As áreas problemáticas sempre foram a ausência de capacidade suficiente de memória e a velocidade relativamente lenta do processamento. Mas os avanços da tecnologia nos últimos anos fizeram com que a qualidade de tais programas dependa, agora, do software.

Como os computadores são calculadoras que operam em alta velocidade, o xadrez para computador é projetado com base em cálculos numéricos, usados para avaliar os dois elementos essenciais do jogo: o material e a mobilidade.

Estratégia em software

Para jogar xadrez contra o computador é preciso, em geral, conhecer as regras do jogo. Isso porque a maioria dos softwares disponíveis, tanto no mercado nacional como no estrangeiro, classifica os jogos de xadrez por níveis.

Os do nível 1 ao nível 8 são elaborados para jogadores de conhecimentos técnicos medianos e compõem a totalidade dos programas de xadrez oferecidos no mercado nacional. Existem jogos de níveis superiores, importados, que incluem lances de campeonatos internacionais, e alguns até acima do nível de disputa entre campeões.

Entre os programas nacionais, destacam-se dois jogos de xadrez Microsoft, desenvolvidos para micros TK 83/85 e TK 2000.

Para utilizar um jogo de xadrez no TK 2000 Color, é necessário um gravador mono e um cabo ligando o monitor, pela saída EAR. Depois de passar a fita com o programa para a memória do micro, surgirão na tela o tabuleiro e as instruções para iniciar o jogo.

Conhecendo-se as regras, o jogo da Microsoft é muito simples. De início, o computador apresenta na tela o tabuleiro com as peças colocadas. (Alguns softwares estrangeiros, como o Colossus, não apresentam esse recurso.) As peças do jogador encontram-se na parte inferior do tabuleiro, mas essa

O material de um jogo refere-se à quantidade e ao valor das peças no tabuleiro. O programa de xadrez atribui a cada peça um valor numérico. O rei pode receber um valor infinito ou um valor arbitrário alto, como 10.000 (isso porque a perda iminente dessa peça encerra o jogo); a rainha recebe o valor nove; a torre vale cinco; os bispos e os cavalos, três; e os peões, um.

Para decidir se vale a pena sacrificar uma peça a fim de capturar uma das peças do oponente, o programa compara seus valores. A maioria dos programas de xadrez para computador dá grande importância aos valores relativos e raramente trocará peças se isso resultar em desvantagem material, a menos que haja um avanço acentuado de posição.

A mobilidade é fator muito importante no xadrez: qualquer peça terá pouco valor se seu movimento estiver limitado. Inversamente, seu valor aumentará se ela puder ser posicionada de forma que assegure influência em várias áreas do tabuleiro ao mesmo tempo. O programa de xadrez precisa, desse modo, avaliar a mobilidade, assim como os aspectos materiais. Além disso, o programa deve planejar com antecedência, determinando a melhor seqüência de movimentos a partir de qualquer posição.

É aí que os programas de xadrez se mostram notáveis, usando a velocidade de processamento do computador para examinar um grande número de movimentos possíveis em muito menos tempo do que despenderia uma pessoa.

A maioria dos programas de xadrez usa uma técnica de "força bruta", pesquisando o maior número de movimentos possível no tempo permitido. A duração de cada movimento é determinada pela escolha do nível de jogo, cada um dando um espaço de tempo diferente durante o qual o computador deve fazer um movimento.

Esses períodos variam de alguns segundos até várias horas e, quanto mais tempo tiver para a pesquisa, tanto mais probabilidade o computador terá de encontrar a melhor linha de ataque para a posição atual.

A cada movimento, o computador verifica se o rei está ou não em xeque e a seguir examina que probabilidade as peças têm de ser ganhas ou perdidas, se há casas importantes que podem ser ocupadas e outras questões semelhantes. Quanto mais critérios o programa examinar, melhor será o resultado. A questão final é a de descobrir se o rei do oponente pode ser forçado a uma posição de xeque-mate.

Nos jogos entre computadores e seres humanos, a máquina tem acentuada vantagem em termos de velocidade e amplitude de pesquisa — mesmo assim, um excelente jogador humano sempre vence um excelente programa de xadrez para computador, devido a sua capacidade de visualizar e criar novas aberturas e posições. Os

posição pode ser invertida teclando-se [X] seguido de [Return]. O jogador escolhe o nível de jogo digitando [QI] = [1] ou qualquer algarismo até 8. Se você digitar [N], o computador mostrará as coordenadas dos quadrados do tabuleiro, utilizando, na horizontal, as letras de A a H e, na vertical, os números de 1 a 8.

Para movimentar as peças, é necessário digitar as coordenadas que indicam sua posição inicial seguida das coordenadas da posição final; depois, basta digitar [Return] para confirmar a jogada. Por exemplo, digitando-se [B5-B7], a peça que está no quadrado B5 move-se para o B7. Erros de digitação podem ser corrigidos por meio da tecla [<]. Se algum dos movimentos for ilegal, o computador acusará com um ponto de interrogação e um bip, exigindo nova jogada. O jogador pode relegar ao computador a tarefa de jogar por si digitando [P] seguido de [Return]. O programa permite ainda o roque, que é executado sem consulta quanto à legalidade. Para realizar um roque pequeno, digita-se [0-0]; para um roque grande, [0-0-0], sempre seguidos de [Return].

É possível ainda analisar cada situação de tabuleiro, o que se faz acrescentando um asterisco ao final de cada movimento. Nesse caso, o computador não testa a legitimidade do movimento.

Características	Grand Master 64	Colossus	Cyrus IS	Sargon III
Jogo invisível	NÃO	SIM	NÃO	NÃO
Lista os movimentos	NÃO	SIM	NÃO	NÃO
Reapresenta um jogo	NÃO	SIM	SIM	SIM
Mostra a listagem	NÃO	SIM	NÃO	SIM
Imprime a listagem	NÃO	NÃO	SIM	SIM
Retoma posições	SIM	SIM	SIM	SIM
Treinamento	SIM	NÃO	NÃO	SIM
Promove peão	NÃO	SIM	SIM	SIM
Imita o jogo humano	NÃO	SIM	SIM	SIM
Muda de lado	SIM	SIM	SIM	SIM
Analisa problemas	NÃO	SIM	SIM	SIM
Mostra pesquisa	LANCE ÚNICO	SIM	NÃO	SIM
Inverte o tabuleiro	SIM	SIM	SIM	SIM
Oferece empate	NÃO	NÃO	NÃO	SIM
Imprime o tabuleiro	NÃO	NÃO	SIM	SIM
Grava o jogo	NÃO	SIM	SIM	SIM
Relógio de tempo real	SIM	SIM	NÃO	SIM

Xadrez internacional

Um bom programa de xadrez inclui roque, promoção de peão, captura **en passant**, e compreende situações de empate e de xeque perpétuo. O quadro acima mostra algumas das possibilidades de quatro jogos de xadrez para computador disponíveis no mercado internacional.

computadores jogam um excelente xadrez tático, mas, mesmo entre os mestres humanos do xadrez, o jogador que se posiciona bem costuma vencer o bom jogador tático.

Os programadores de xadrez para computador concentraram-se na tática porque, para o computador, o jogo tático envolve apenas lidar com números. Se um opositor humano faz um movimento não convencional, o computador quase sempre falha na escolha da melhor resposta. Por isso, muitos programas de xadrez têm dificuldade em lidar com posições estáticas, em que nenhum dos movimentos possíveis oferece vantagem tática específica. Nessas situações, o programa muitas vezes apenas muda as peças de lugar, em vez de aproveitar a oportunidade de planejar sua estratégia.

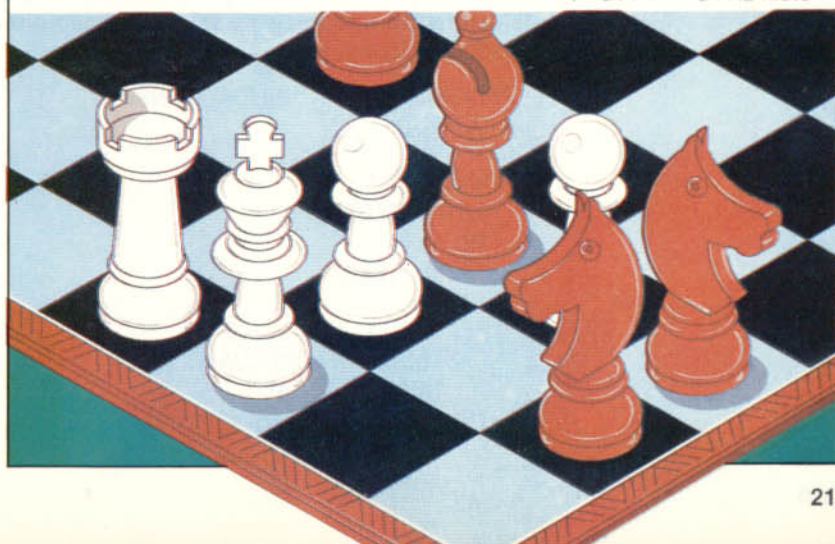
Um estilo de programação recentemente desenvolvido recorre à pesquisa seletiva. Usando essa técnica, o computador imita o jogador humano: examina com maior atenção um número menor de movimentos possíveis. O programa Mephisto III utiliza a técnica de pesquisa seletiva para os dois primeiros movimentos; depois, restringe a pesquisa e examina um conjunto menor de movimentos em detalhe. O Mephisto III também tenta distinguir entre posições táticas e estáticas — técnica que, acredita-se, tornará os computadores do futuro um verdadeiro desafio aos jogadores humanos.

Coração x cabeça

A capacidade de examinar todas as posições com até nove movimentos de antecipação quase garante aos programas de xadrez uma superioridade tática com relação aos seres humanos. A técnica dos mestres enxadristas consiste em selecionar alguns movimentos decisivos nos quais concentram enorme capacidade de análise, com até trinta movimentos de antecipação.

A ilustração mostra um movimento decisivo do jogo de Moritz (pretas) contra Emmerich (brancas), em 1922. As pretas podem dar xeque-mate sacrificando a rainha e fazendo três elegantes movimentos com o cavalo; a maioria dos jogadores preferiria, sem hesitar, essa seqüência. O próprio Moritz não a percebeu e lamentou amargamente seu descuido. Os programas chegariam ao mate, sem, contudo, sugerir os lances com os cavalos.

A seqüência dos movimentos:

1		H5-H2 xeque
2	G1-H2	E5-G4 xeque
3	H2-G1	F4-H3 xeque
4	G1-F1	G4-H2 mate

1 + 2 = 20?

A aplicação dos vários tipos de dados a cálculos aritméticos e o controle de loops do programa por estruturas iterativas são os recursos da linguagem PASCAL que examinaremos em pormenores neste artigo.

A afirmação de que um mais dois é igual a três pressupõe que estamos lidando com números naturais puros. Mas, quando se utiliza uma nota de 1.000 cruzeiros e duas de 500, o resultado da operação, sob outro ângulo, pode ser um maço de cigarros. Esse tipo de classificação de dados é fundamental na descrição de qualquer problema e o PASCAL dispõe de vários recursos para organizá-los e descrevê-los de modo claro e lógico.

A criação de novas categorias de dados torna possível a solução do problema, facilitando o projeto dos algoritmos com base no processamento adequado a cada categoria. Se, por falta de atenção, tentarmos fazer algo sem sentido — por exemplo, acessar um valor booleano por meio do teclado —, o compilador PASCAL interceptará de imediato esse erro lógico. Muito trabalho é poupado evitando a execução de qualquer instrução antes que todos os erros tenham sido eliminados do programa-fonte.

Para maior proveito das descrições de dados "fortemente tipificadas", o usuário do PASCAL pode impor algumas restrições adicionais às variáveis. Em resumo, as regras para manipulação dos tipos simples de variáveis são: nenhum dos escalares é "compatível" com variáveis de qualquer outro tipo escalar, embora os dois tipos numéricos possuam muitas características comuns; e, assim como os números reais podem ser arredondados para inteiros, é possível a atribuição de valores do tipo integer a reais, mas não o inverso. Eis alguns exemplos:

```
PROGRAM COMPATIBILIDADE (INPUT,
  OUTPUT);
VAR
  INTA,
  INTB :INTEGER;
  XREAL,
  YREAL :REAL;
BEGIN
  READ (INTA, XREAL); {LE UM
    INTEGER, DEPOIS QUALQUER
      NUMERO LEGAL}
  YREAL:=INTA; {REAL:=INTEGER
    E CORRETO}
  INTB:=XREAL; {** ERRO : ILEGAL **}
  {..ETC.}
```

As operações aritméticas são definidas apenas nos tipos numéricos. Tanto para os valores integer como para os reais podem-se usar os quatro operadores simbólicos habituais:

+ (adição);
– (subtração);
* (multiplicação);
/ (divisão com vírgula flutuante).

Nesse contexto, tornam-se operadores "diádicos" ou binários, pois sempre exigem dois "operandos" de qualquer um dos tipos numéricos. Toda vez que um dos operandos for real, o resultado da expressão também o será, de modo que: 2+2,0 serão 4,0 (e não 4). No caso do símbolo de divisão, a expressão sempre resultará em um número real, mesmo quando os dois operandos forem integers: 3/5 será 0,6, 8/4 serão 2,0.

Na divisão de valores inteiros, muitas vezes a resposta "real" não faz sentido. Por exemplo, doze barras de chocolate divididas entre dez pessoas resultarão em uma para cada pessoa, restando duas barras. O resultado integer e o resto podem ser obtidos com os dois operadores de divisão integer DIV e MOD (ambos palavras reservadas do PASCAL): 15 DIV 5=3 e 15 MOD 5=0; 31 DIV 7=4 e 31 MOD 7=3.

Preste atenção para a possibilidade de os dados serem negativos, pois não se pode realizar uma divisão integer quando o denominador assume valores negativos. Na execução do programa, qualquer tentativa de dividir por zero resultará numa mensagem de erro — pelo menos enquanto não se inventar um modo de avaliar o infinito.

Em toda expressão, as operações de multiplicação e divisão são efetuadas antes das de adição e subtração. Emprega-se a notação com parênteses para inverter essa "prioridade" lógica. Por exemplo, (8+4) DIV 2=6, mas 8+4 DIV 2=10.

Funções de transferência

Embora não possamos fazer uma atribuição direta de um valor real a uma função integer, o PASCAL possui "funções de transferência" muito úteis para isso — os identificadores TRUNC e ROUND. A função TRUNC elimina (trunca) a parte fracionária de um número real, qualquer que seja ela; por exemplo, TRUNC (3,999)=3 e TRUNC (–123,456)= –123.

A função ROUND executa um arredondamento "inteligente": para mais se a parte fracionária for maior que 0,5, e para menos em caso

contrário. Então, ROUND (1.234,6) = 1.235 e ROUND (-0,49237) = 0.

Ocorrerá erro se o argumento real para qualquer dessas funções resultar num valor inteiro fora da faixa do tipo integer (-MAXINT a MAXINT); assim, sempre verifique antes utilizando a instrução IF. Do mesmo modo, o argumento precisa ser real e não integer, pois já terá sido truncado e arredondado.

O PASCAL possui a função ODD (ímpar), que fornece um resultado booleano — falso se seu argumento integer for um número par, e verdadeiro se for ímpar:

```
IF ODD (N)
  THEN
    WRITELN ('E IMPAR!')
  ELSE
    WRITELN ('ISSO TORNA PAR!')
```

ou:

```
IF ODD (N) THEN
  IF N MOD 2 = 0 THEN
    WRITELN ('CONSERTE O COMPILADOR!')
```

A tabela da página seguinte mostra todas as funções aritméticas do PASCAL. Elas podem assumir qualquer argumento numérico escalar, real ou integer. As duas primeiras, ABS e SQR, fornecerão um valor de tipo compatível com seu argumento. Assim, ABS (-19,372) será 19,372 e ABS (255) será 255.

As outras funções, porém, sempre fornecerão um resultado real, de modo que SQRT (16) será 4,0 e não 4. Isso porque os algoritmos usados no cálculo das funções baseiam-se na soma de uma série de termos, todos fracionários. Nas referências aos resultados reais das funções aritméticas, deve-se dizer "será o mesmo que" em vez de "igual a", pois não é seguro comparar valores reais pela igualdade exata. O menor erro nos cálculos significará que dois valores reais nominalmente "iguais" podem de fato diferir por, digamos, 1,0E-27. Erro insignificante, mas o computador não sabe disso. Em vez de usar IF X = Y THEN.., podemos usar a função predefinida do PASCAL ABS:

```
IF ABS (X-Y)<QUASE ZERO THEN {ETC.}
```

Quando a diferença entre X e Y for muito pequena, recomendamos o seguinte teste. A aproximação considerada desprezível pode ser especificada com proveito mediante uma definição CONST no início do programa. Os logaritmos são fornecidos na base natural (e), não 10, e exp eleva e à potência de seu argumento.

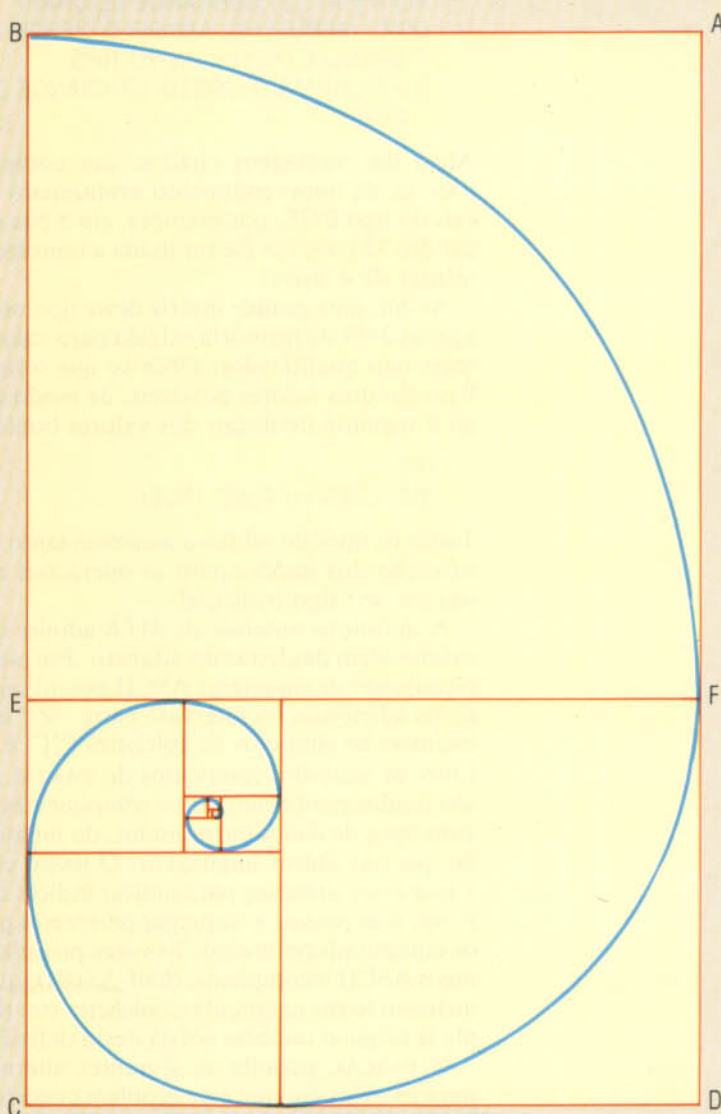
Como o PASCAL não possui operador de potenciação, evitam-se certas preposições absurdas muito comuns no BASIC, como:

```
500 LET D=B^2+4 * A * C
```

que calcula o quadrado de um número por um procedimento muito mais lento e impreciso do que a simples instrução B * B. Observe que a função SQR (N) do PASCAL fornece um valor in-

A divisão áurea

Um e somente um ponto corta uma linha em duas partes, de tal modo que a razão entre seus comprimentos é igual à razão entre a parte maior e a linha inteira. Essa razão — chamada áurea — é um número irracional como o pi, com dígitos até o infinito.

A divisão áurea, usada pelos artistas e arquitetos gregos, é freqüentemente retomada nas teorias da proporção, mas suas manifestações mais comuns se encontram na natureza, como em conchas marinhas: os vetores de seus raios, separados em 90°, correspondem à razão áurea.

As arestas do retângulo ABCD respeitam a razão áurea. Traçando-se um quadrado ABEF, o novo retângulo EFDC tem arestas na mesma razão que ABCD. Repetindo-se o processo, os vértices dos quadrados continuam sendo pontos da "espiral áurea".

teger para o quadrado de N (supondo-se que N seja integer), enquanto SQR (X/3) forneceria um valor real. Outra diferença do PASCAL em relação ao BASIC: emprega-se a função SQRT (ALGO), e não SQR (ALGO), para se obter a raiz quadrada de um número.

Tipos de subfaixa

Cada tipo escalar possui uma faixa (range) definida e ordenada de valores, podendo ser usado como um tipo "principal" do qual derivamos tipos secundários definidos por subfaixas (subranges) de valores. Os limites inferior e superior de cada subfaixa são indicados por dois pontos (..) na definição do identificador do tipo de subfaixa; por exemplo:

```
TYPE
  BYTE = 0..255; {SUBRANGE DE INTEGER}
  ALFA = 'A'..'Z'; {SUBRANGE DE CHAR}
  COR = (VERMELHO, AMARELO, VERDE,
    BRANCO, PRETO); {NOVO TIPO}
  B e P = BRANCO..PRETO; {SUBRANGE DE
    COR}
```

Além das vantagens citadas, um compilador PASCAL de bom rendimento armazenará variáveis do tipo BYTE, por exemplo, em 8 bits em lugar dos 32 possíveis (se for usada a representação integer de 4 bytes).

Assim, uma grande matriz desse tipo ocupará apenas 25% da memória exigida para valores integer não qualificados. Observe que o tipo B e P possui dois valores possíveis, de modo análogo à seguinte definição dos valores booleanos:

```
TYPE
  BOOLEAN = (FALSE, TRUE);
```

Todos os tipos de subfaixa assumem tanto a classificação dos dados como as operações definidas em seu tipo principal.

A definição anterior de ALFA admite outros valores além das letras do alfabeto. Em especial, o conjunto de caracteres ASCII possui seis símbolos adicionais no intervalo entre "Z" e "a", inclusive os símbolos de colchetes ("[" e "]"). Entre os símbolos reservados do PASCAL, esses são usados para delimitar os componentes de alguns tipos de dados estruturados, do mesmo modo que nas outras linguagens. O BASIC chegou a usar esses símbolos para indicar índices de matrizes, mas passou a empregar parênteses porque os computadores antigos às vezes possuíam códigos ASCII incompletos, (half-ASCII), que não incluíam letras minúsculas, colchetes etc. (O Apple II original também sofria dessa deficiência.)

O PASCAL permite as seguintes alternativas para os casos em que esse problema ocorre: (. e .) em vez de [e], e (* e *) no lugar de {e}. Para delimitadores de comentários, a segunda opção é bastante comum; no entanto, apenas os compiladores de padrão ISO permitem a substituição das chaves. Apenas uma ou as duas substituições podem ser feitas, de modo que: (* ESTE E UM COMENTARIO VALIDO}.

Função	Valor resultante
ABS (K)	Valor absoluto de K
SQR (K)	Quadrado de K
SQRT (K)	Raiz quadrada de K
SIN (A)	Seno de A radianos
COS (A)	Co-seno de A radianos
ARCTAN (T)	Ângulo com a tangente T
LN (K)	Logaritmo natural de K
EXP (L)	"e" elevado à potência L

Programa Áureo

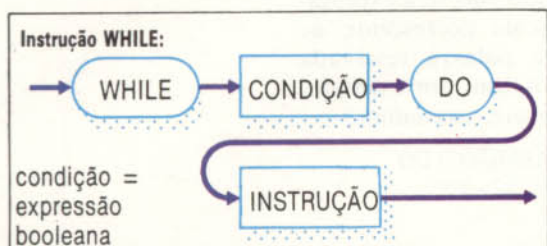
O programa Áureo, abaixo, produz termos de Fibonacci e os imprime juntamente com sua razão. Ele revela uma das propriedades interessantes da série de Fibonacci (1, 2, 3, 5, 8, 13 etc.), a de que cada par sucessivo de termos se aproxima gradativamente do valor da divisão áurea.
O programa fornece exemplos adicionais da estrutura REPEAT do PASCAL. Como exercício, tente modificar a condição de encerramento do loop para que ele se interrompa quando o próximo termo de Fibonacci a ser calculado exceder MAXINT.

```
PROGRAM     AUREO     ( OUTPUT );
CONST
      EPSILON      = 1.0E - 08;
TYPE
      FIBONACCI    = 1 .. MAXINT;
VAR
      PRIMEIRO,
      SEGUNDO,
      PROXIMO      : FIBONACCI;
      RAZAO,
      OURO         : REAL;
      CONTADOR     : INTEGER;
BEGIN
  WRITELN ( 'A DIVISAO AUREA' : 30 );
  WRITELN;
  WRITELN ( 'SERIE DE FIBONACCI :' );
  PRIMEIRO := 1; { POR DEFINICAO }
  SEGUNDO := 1;
  RAZAO := PRIMEIRO/SEGUNDO
  CONTADOR := 0;
  REPEAT
    IF CONTADOR MOD 10 = 0 THEN
      BEGIN { CABECALHO A CADA 10 LINHAS }
        WRITELN;
        WRITELN ( 'PRIMEIRO' : 10, 'SEGUNDO' : 10,
          'RAZAO' : 14 );
        WRITELN;
      END;
    WRITELN ( PRIMEIRO : 10, SEGUNDO : 10,
              RAZAO : 16 : 8 );
    CONTADOR := CONTADOR + 1;
    OURO := RAZAO; { SALVA A RAZAO
      ANTERIOR }
    PROXIMO := PRIMEIRO + SEGUNDO;
    PRIMEIRO := SEGUNDO { AVANCO }
    SEGUNDO := PROXIMO; { A SERIE }
    RAZAO := PRIMEIRO / SEGUNDO
  UNTIL ABS ( RAZAO — OURO ) < EPSILON;
  WRITELN;
  WRITELN ( 'A RELACAO AUREA E : ',
          100 * RAZAO : 10 : 5, '%' );
END.
```

Estruturas iterativas

Além das estruturas seqüenciais, que empregam BEGIN e END, e da escolha ou seleção, que utiliza IF e CASE, o PASCAL dispõe de um terceiro tipo de instrução: a "iteração", ou repetição. A instrução FOR resulta num loop "controlado por contador", enquanto as instruções WHILE e REPEAT são loops "controlados por condições".

Em primeiro lugar, calcula-se a expressão booleana delimitada pelas palavras reservadas WHILE e DO. Se o resultado for verdadeiro, a instrução seguinte do PASCAL (qualquer uma, mesmo as estruturadas, independentemente de sua complexidade) será executada repetidas vezes, enquanto a condição permanecer verdadeira.

Essa condição booleana é reavaliada após cada execução da instrução no "núcleo" da estrutura WHILE. Isso significa que pelo menos um

dos itens avaliados na expressão deve receber alguma modificação pelas atividades no interior do loop. Por exemplo, a instrução

```
WHILE MAXINT > 1 DO
  WRITELN ('EM LOOP!')
```

teria grande dificuldade em se encerrar. Eis um exemplo prático (e mais confiável):

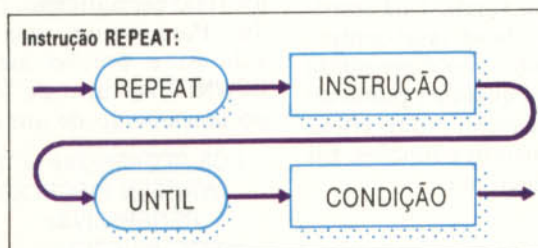
```
READ (RETIRADA);
WHILE RETIRADA > SALDO DO
  BEGIN
    WRITELN ('SEM FUNDOS, TENTE
      NOVAMENTE');
    WRITE ('QUANTIA?');
    READ (RETIRADA)
  END
```

A mais importante vantagem da estrutura WHILE é bem exemplificada por esse fragmento de programa. Se, quando for lido pela primeira vez, o montante sacado estiver dentro do limite do saldo atual da conta corrente, o loop WHILE não será executado. Se o núcleo for executado, o loop será repetido até que a expressão se torne falsa. O fluxo de controle retornará à operação seqüencial, e a primeira instrução após a estrutura WHILE será então executada. Assim, para realizarmos uma iteração em que apenas uma condição é verdadeira, usamos o loop WHILE.

Muitas vezes ocorrem situações em que precisamos executar o núcleo de um loop pelo menos uma vez para estabelecer a condição que o interromperá, como acontece no exemplo do programa Áureo. Nessas circunstâncias, a estrutura REPEAT-UNTIL traduz o algoritmo com maior precisão e fornece uma sintaxe natural de alto nível.

Observe que a condição de encerramento agora aparece no final da estrutura e que ela é oposta, em termos lógicos, à condição usada para codificar a estrutura WHILE. Isso significa que o loop REPEAT se encerrará quando a expressão booleana resultar verdadeira, enquanto uma estrutura WHILE se interromperá (ou sequer se iniciará) se a condição for falsa.

Se o núcleo da instrução REPEAT contiver instruções, elas deverão ser escritas entre as palavras delimitadoras BEGIN-END, como uma instrução composta; mas essa regra não é estrita, porque as palavras reservadas REPEAT e UNTIL delimitam adequadamente o núcleo do loop. Nada impede que se introduzam as palavras supérfluas BEGIN e END (procedimento parecido com o uso de ponto-e-vírgula antes de uma pa-

lavra reservada), embora isso não crie nenhuma instrução nula.

O fragmento abaixo conta a freqüência da letra "e" numa sentença fornecida por meio do teclado. Você deverá finalizar a sentença com um ponto quando o loop REPEAT se encerrar, e o resultado será apresentado.

```
CONTADOR: = 0;
REPEAT
  READ (SIMBOLO);
  IF SIMBOLO = 'E' THEN
    CONTADOR: = CONTADOR + 1
UNTIL SIMBOLO = '.';
WRITELN ('EXISTEM', CONTADOR : 1,
  ' "E"S NA SENTENCA.')
```

Nesse caso, a prioridade de qualquer das estruturas não é muito acentuada, e um loop WHILE poderia ser igualmente usado:

```
CONTADOR: = 0;
READ (SIMBOLO);
WHILE SIMBOLO < > '.' DO
  BEGIN
    IF SIMBOLO = 'E' THEN
      CONTADOR: = SUCC (CONTADOR);
    READ (SIMBOLO)
  END;
WRITE ('EXISTEM', CONTADOR : 1,);
WRITELN (' "E"S NA SENTENCA.')
```

É fácil perceber a diferença entre as duas estruturas. O procedimento necessário para estabelecer a condição de encerramento (a instrução READ, no caso) em geral ocorrerá duas vezes com o uso da estrutura WHILE: primeiro, antes

da entrada para a estrutura; depois, como a última instrução no núcleo do loop WHILE.

Embora o loop WHILE seja a única estrutura essencial em qualquer linguagem, convém ter um meio de repetir algo um número especificado de vezes, talvez numa certa faixa de valores no interior de uma escala. O PASCAL possui uma terceira estrutura para criar esses loops "controlados por contador". Nesse ponto, o programador em BASIC se sentirá em terreno conhecido, mas há várias diferenças importantes entre a estrutura do PASCAL e o loop FOR do BASIC.

Em primeiro lugar, qualquer variável escalar pode ser usada como controlador do loop e não apenas as variáveis integer. Além disso, a instrução FOR-DO do PASCAL é completamente confiável; tal como o loop WHILE, ela não poderá ser executada de forma alguma (se, por exemplo, o valor inicial for maior que o final), e há restrições drásticas quanto ao uso da variável controladora no PASCAL. Sobretudo, não é permitida a modificação desse valor em qualquer ponto do núcleo do loop — uma precaução muito eficaz quando se lida com procedimentos e funções. Eis um exemplo de estrutura ilegítima:

```
FOR N:=1 TO 10 DO
  IF N=10 THEN
    N:=1
```

Isso não faz sentido, pois, apesar de se determinar que o loop seja executado dez vezes, a instrução IF restabelecerá o valor 1 para N, criando assim um loop infinito.

Observe que a sintaxe da estrutura FOR-DO possui uma instrução de atribuição entre os dois primeiros delimitadores (FOR e TO), que atribui o valor inicial à variável de controle do loop. Já o valor final da escala é dado pela expressão escrita entre as palavras reservadas TO e DO. Esses dois valores devem ser do mesmo tipo escalar simples que o controlador. Eis outros exemplos:

- FOR LETRA:='A' TO 'Z' DO |ETC|
- FOR MES:=JAN TO DEZ DO |ETC|
- FOR N:=N TO SUCC (MAXINT DIV 1000) DO |ETC|

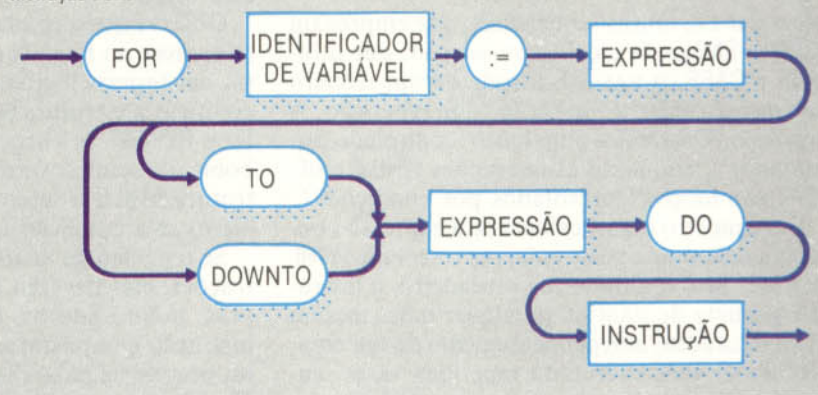
Resposta do exercício áureo

Com a representação do complemento dos pares, se um resultado integer ultrapassar MAXINT, o bit de sinal ficará "ligado". O número então aparece negativo; desse modo, nosso programa Áureo não funcionaria, já que o tipo Fibonacci exclui valores negativos. Mesmo com valores integer, o loop nunca se encerraria. Não podemos dizer:

```
UNTIL PRIMEIRO+SEGUNDO>MAXINT
```

pois, por definição, não há valor maior que MAXINT. Contudo, uma reestruturação engenhosa para efetuar a subtração em vez da adição permite resolver isso:

```
UNTIL SEGUNDO>MAXINT — PRIMEIRO
```

Agora a aritmética real e a constante épsilon já não são mais necessárias.

O último exemplo supõe que N tenha recebido um valor adequado em algum ponto anterior do programa, mas, se esse valor ultrapassar o valor final especificado, o loop não será executado. Para seguir uma escala decrescente de valores, é preciso usar a palavra reservada DOWNTO no lugar de TO. Assim, por exemplo, no lançamento de um foguete, usaríamos:

```
FOR REGRESSIVA:=10 DOWNTO 0 DO
  WRITELN (REGRESSIVA : 32-3*
    REGRESSIVA);
WRITELN ('FOGO!')
```

O PASCAL mantém um controle rigoroso do loop FOR e não admite qualquer equivalente do incremento ou decremento por meio do comando opcional STEP do BASIC.

```
PROGRAM      CERVEJA      (OUTPUT);

TYPE
  TIPO = (LEVE, AMARGA, ESPECIAL, PRETA);
VAR
    PRECO,
    GARRAFAS : REAL;
    CERV : TIPO;
BEGIN
  WRITELN ( 'GARRAFAS' : 11,
       'SUAVE' : 9, 'AMARGA' : 9,
         'ESPECIAL' : 9, 'PRETA' : 9);
  WRITELN;
  GARRAFAS :=0.5; | COMECA COM MEIA |

  REPEAT
    IF GARRAFAS - TRUNC ( GARRAFAS ) > 0.4
      THEN | IMPRIME COMO XX.X |
        WRITE ( GARRAFA : 10 : 1 );
      ELSE | IMPRIME COMO INTEIRO |
        WRITE ( ROUND ( GARRAFAS ) : 7, ' ' : 3 );

    FOR CERV := SUAVE TO PRETA DO
      BEGIN
        CASE CERV OF
          SUAVE : PRECO := 1000;
          AMARGA : PRECO := 1200;
          ESPECIAL : PRECO := 1500;
          PRETA : PRECO := 1300;
        END;
          | ARREDONDA E IMPRIME O PRECO |
        WRITE ( ROUND ( PRECO * GARRAFAS ) : 9 : 2 );
      END;
        | INICIA NOVA LINHA |
    WRITELN
    GARRAFAS := GARRAFAS + 0.5;

  UNTIL GARRAFAS > 10;
END.
```

Qual é a sua?

O programa Cerveja imprime uma lista com os preços de quatro tipos de cerveja, numa escala que aumenta em graus de meia garrafa. Cada tipo de cerveja é representado por um identificador de constantes conceituais enumeradas na definição TYPE. A seleção do preço é feita por uma instrução CASE. As garrafas inteiras são impressas em valores integer, com dois espaços adicionais, enquanto a formatação dos totais reais elimina casas decimais não desejadas.

WAFADRIVE

O aparecimento do sistema Microdrive da Sinclair estimulou companhias independentes a produzir equipamentos para armazenamento de dados, tais como o Wafadrive da Rotronics.

Uma vez conectado à CPU do Spectrum ou do TK 90X, o Wafadrive agrupa várias interfaces e o dispositivo de armazenamento, além de slots para a ligação de outros periféricos.

O nome dado a unidades desse tipo, "floppy tape", indica sua posição entre as fitas cassete e os discos flexíveis. O princípio de seu funcionamento é simples: o rolo de fita é contínuo, de modo que rapidamente se encontram os dados. E, como o diretório dos discos, o floppy tape mantém um índice de todos os programas e arquivos armazenados na fita. Dessa forma, uma lista do conteúdo está sempre à disposição do usuário.

O Wafadrive é envolto numa caixa de plástico preto com cabo flexível multivias finalizado por um conector que se encaixa no bus de expansão do microcomputador. Na parte frontal, há duas janelas para as unidades de minicartuchos. Entre as janelas estão três diodos de emissão de luz (LEDs). O LED central é o indicador de ligado/desligado; os outros dois indicam se a unidade está em funcionamento.

Fácil acesso
Ao contrário do Microdrive da Sinclair, o Wafadrive vem num estojo único com dois acionadores de fita, uma saída serial RS232 e uma interface paralela Centronics. O cabo flexível adapta-se ao conector edge e com isso a unidade pode ficar logo atrás do teclado, facilitando o acesso.

Isso significa que o Wafadrive agrupa numa só unidade várias interfaces e o dispositivo de armazenamento. Possui também slots para ligação de outros periféricos.

Atrás da unidade existem três conectores tipo edge. À esquerda há um para o bus paralelo. O conector edge central é uma interface compatível com o padrão Centronics e dá saída para uma impressora paralela. O terceiro corresponde a uma saída serial padrão RS232 que possibilita a utilização de modems e outros dispositivos seriais.

Os minicartuchos projetados especificamente para o Wafadrive assemelham-se aos usados nos Microdrives da Sinclair. Dentro do cartucho existe uma fita contínua do tipo usado para videocassete, com largura de 1,8 mm. Ela é usada no lugar das fitas de áudio convencionais porque tem maior resistência e maior capacidade de armazenamento de informações. Uma vez formatada, a fita pode conter até 128 Kbytes de dados.

Os cartuchos são quase duas vezes mais largos que os produzidos pela Sinclair, se bem que de comprimento e espessura similares. Não precisam ser acondicionados em caixas protetoras, pois uma cobertura deslizante automática, parecida com a existente nos minidisquetes de 3 1/2 polegadas da Sony, reveste a fita. No lado esquerdo do cartucho há uma plaquinha de proteção contra gravação.

Os comandos do Wafadrive são bem semelhantes aos usados pelo Microdrive da Sinclair. Nos dois sistemas o comando é seguido por um *, como em SAVE *, LOAD * e VERIFY *, indicando que o dispositivo de armazenamento externo deve ser acessado. Quando se formata um cartucho da Sinclair usa-se o comando FORMAT 'm' ;0;"nome", onde 'm' ;0 se refere ao número do Microdrive em operação. Quando se usa o Wafadrive, o comando é FORMAT 'a:nome', onde a: se refere ao nome da unidade em uso. Com o Wafadrive, só pode haver a designação a: ou b:; no Microdrive o número pode variar de zero a sete.

O sistema stream

O Wafadrive também tira vantagem do sistema stream usado no Spectrum, no qual existem dezesseis canais reservados para controle de entrada/saída. Alguns desses streams são ocupados pela tela e pela impressora. Os canais de números 4 a 15 ficam disponíveis para outros periféricos e os streams de saída para o Wafadrive são acessados pelo comando OPEN #. O Wafadrive também adiciona dois streams extras ao sistema: os canais R e C, reservados para a saída serial RS232, e a paralela Centronics, respectivamente.

Uma ROM de 8 Kbytes, na placa principal, contém os comandos ampliados do BASIC usados para o controle do sistema. Esse sistema de operação do Wafadrive (WOS, Wafadrive Operating System) funciona paginando os 8 Kbytes inferiores da ROM do Spectrum. O comando LOAD *, por exemplo, gera um erro; dessa forma, quando o interpretador BASIC encontra tal comando na tela, chama uma rotina que se encarregue desse erro. O comando, no entanto, é interceptado pelo WOS, que aciona a ROM do Wafadrive. Esta, por sua vez, assume a manipulação do erro e interpreta LOAD * como comando.

Comparado com o Microdrive, o Wafadrive da Rotronics é um pouco lento. Um Microdrive de 100 Kbytes requer, em média, 3,5 segundos para localizar uma informação, transferida para o computador a uma velocidade de até 19,2 Kbauds. Por outro lado, o Wafadrive só atinge uma velocidade de transferência de 18 Kbauds, com tempo máximo de acesso de 45 segundos num cartucho de 128 Kbytes. Essa menor rapidez é parcialmente compensada por sua mais alta confiabilidade.

Embora esses tempos de acesso sejam muito menores do que os obtidos em fita cassete, são ainda muito maiores que o tempo de acesso das unidades de disco. No entanto, os floppy tapes incorporam um procedimento útil para acessar o catálogo da fita, equivalente ao diretório dos discos e disquetes. O catálogo é armazenado no primeiro setor do cartucho, após a emenda que une as duas extremidades da fita. Desse modo, para se obter o catálogo de um cartucho, o acionador tem de girar a fita até encontrar a emenda e ler o próximo setor. Depois de vários segundos, o cabeçote da fita passa o setor do catálogo. No entanto, se o comando CATALOGUE for usado de novo, o acionador, ao invés de girar a fita inteira mais uma vez, move-se apenas por uma fração de segundo e mostra o mesmo catálogo de novo — uma vez chamado, ele se mantém na RAM.

Portanto, o WOS apenas verifica se é o mesmo cartucho que está inserido, examinando o setor seguinte. Em caso afirmativo, o WOS mostrará o catálogo que ele já incorporou à RAM.

Um pequeno preço?

Seguindo a linha de implementar um sistema de armazenamento de dados que permita o uso do Spectrum e seus compatíveis para aplicações profissionais, a Rotronics inclui na venda do Wafadrive o programa de processamento de texto Spectral Writer. Este é um sistema abrangente, que utiliza o Wafadrive por completo. Funções como a inserção de palavras e a eliminação de linhas inteiras são chamadas usando-se a tecla [Shift].

O que faz o sucesso ou o fracasso de qualquer meio de armazenamento num computador é o suporte de software. Isso ainda constitui séria desvantagem para o Wafadrive, uma vez que nenhum dos fabricantes de software mais importantes está produzindo seus programas em cartuchos Wafadrive. Esse problema, aliás, foi enfrentado também pela Sinclair. Mas existe pelo menos um programa que permite transferir aplicativos comerciais para cartuchos Wafadrive.

Os usuários são forçados a comprar o cassete comercial e o cartucho para transferi-lo, mas este talvez seja um preço pequeno para se obter tempos de acesso muito menores.

Outra dificuldade do Wafadrive está nos conectores edge, na parte traseira da máquina. Como eles não são do tipo padrão, os usuários precisam adaptar as próprias interfaces. Se não, devem procurar periféricos com conexões apropriadas.

LEDs de controle das unidades
Indicam ao usuário que unidade está sendo acessada.

Cabo de conexão
O cabo multivias liga-se ao conector edge do Spectrum. Por meio desse bus, o Wafadrive recebe não só os dados de controle como a alimentação necessária.

Chip de ROM
Esta EPROM de 8 Kbytes contém o sistema operacional do Wafadrive (WOS).

Conector paralelo
Este conector edge permite que outras interfaces compatíveis com o Spectrum sejam adicionadas ao sistema.

Acionadores de fita
O Wafadrive possui duas unidades auto-suficientes.

Motores
Cada unidade tem seu motor elétrico próprio para girar as fitas nos cartuchos.

Saída serial RS232
Permite que o Spectrum se ligue a modems e outros aparelhos de comunicação.

Interface paralela padrão Centronics
Faz a conexão com impressoras.

WAFADRIVE

DIMENSÕES
230 x 110 x 180 mm.

INTERFACES
Saída serial RS232, interface paralela padrão Centronics, conector edge do Spectrum.

FORMATO
Minicartuchos flexíveis com fita em anel sem fim.

CAPACIDADE
Acham-se disponíveis cartuchos de 16, 64 e 128 Kbytes.

VELOCIDADE DE TRANSFERÊNCIA
16 Kbauds.

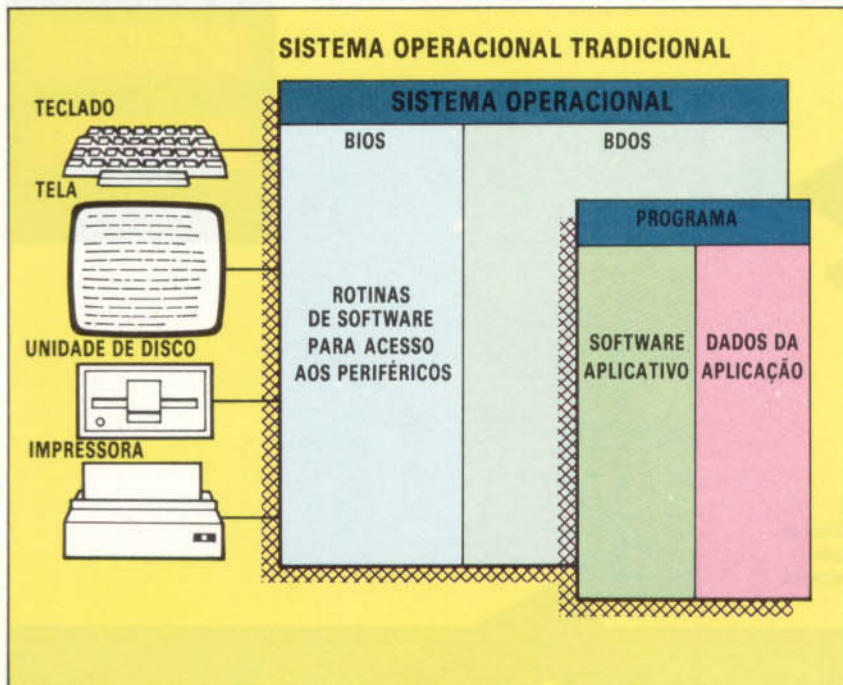
TEMPO MÁXIMO DE ACESSO
6,5 s (16 K), 45 s (128 K).

O minicartucho do Wafadrive tem quase o dobro do tamanho do equivalente do Microdrive da Sinclair. O cartucho assemelha-se a um cassete convencional em suas proporções, mas a fita interna forma um anel sem fim. Isso significa que a fita não precisa ser girada para trás a fim de acessar os dados que já tenham passado pelo cabeçote de leitura e gravação.

PAGINAÇÃO

O termo inglês "paging" traduz-se por "paginação" e indica a divisão de programas ou dados em blocos de tamanho fixo, carregados na memória quando necessário.

CONTROLE TOTAL

A integração mais comum produz programas que abrangem todas as funções de que você necessita. Contudo, esse não é o melhor sistema, pois tais programas são enormes e usam muita memória.

Operando sob ordens
Com o sistema operacional tradicional, o programa em execução tem o controle completo. Sua lógica determina o que aparece na tela, quando é preciso acessar os discos, e o modo de ler o teclado. Suas instruções gerais são transferidas para o sistema operacional, que lida com a manipulação detalhada do hardware em uso. A execução do programa predomina e a subordinação do sistema operacional está pressuposta.

A abordagem alternativa para a questão da integração baseia-se num princípio que conta com o sistema operacional do computador para prover os recursos básicos da integração. Todos os programas escritos para esse sistema ajustam-se automaticamente a ele para trabalharem juntos.

A criação de um sistema operacional desse tipo não foi tarefa fácil, pois ele precisa de um computador com hardware e software muito mais sofisticados do que nos sistemas tradicionais. A Apple abriu o caminho com seus microprocessadores destinados ao Lisa e ao Macintosh e várias outras empresas estão desenvolvendo sistemas para ser rodados em microcomputadores como o IBM PC.

Os programas elaborados para os novos sistemas operacionais são bem diferentes dos desenvolvidos para os sistemas tradicionais. Boa parte desses programas inclui a interface com o usuário, ou seja, as rotinas que recebem comandos e informações do usuário e lhe apresentam os resultados. As opiniões quanto à maneira de operar programas divergem, de forma que cada pacote tem seus próprios procedimentos operacionais exclusivos.

O sistema operacional integrado oferece uma série de rotinas incorporadas para comunicação com o usuário, utilizável em todos os programas aplicativos. Quando precisa mostrar ao usuário uma lista de opções, o programa tem à disposição uma rotina pronta para isso, no sistema operacional. A vantagem é que todos os programas projetados para trabalhar com esse sistema operacional terão aproximadamente o mesmo procedimento. Depois que você tiver aprendido um dos programas do sistema, estará pronto para usar todos os outros à disposição.

Um modo de interagir com o usuário, específico desses programas, é o mouse, dispositivo usado para escolher opções na tela por meio de movimentos correspondentes do cursor.

A alternativa para o mouse é a tela digital, em que uma matriz de raios luminosos responde ao toque do dedo. A tela divide-se em janelas, cada qual contendo uma opção ou uma tarefa diferente.

Recursos como esse exigem um microprocessador rápido, muita memória e alta resolução gráfica. Mas vale a pena arcar com essas despesas, pois o sistema se aplica a quase todos os programas disponíveis. Além disso, é muito fácil de aprender e garante que o usuário visualize várias aplicações diferentes ao mesmo tempo e mude de uma para outra da forma mais simples possível.

Controle de operações

O programa e o usuário nunca estão em contato direto; é o sistema operacional que controla tudo, o tempo todo. Cada aplicativo torna-se uma extensão do sistema operacional e o computador fica sendo um único ambiente integrado.

Isso leva à segunda grande diferença na forma pela qual os sistemas funcionam. Na maneira tradicional, a comunicação entre o sistema operacional e o programa ocorre numa única direção. O programa determina que certa tarefa seja realizada e o sistema operacional a executa. Num processo integrado, o sistema operacional está no comando e atribui tarefas ao programa. O sistema operacional manda, por exemplo, esta mensagem ao programa: "O usuário acaba de mover seus dados para o outro lado da tela. Redesenhe a tela", ou "Pare tudo. O usuário moveu o mouse para uma aplicação diferente". É preciso que o programa seja capaz de responder às requisições do sistema operacional e vice-versa.

Uma vez atingido esse grau de cooperação entre todos os programas, é fácil construir um ambiente integrado. Cada programa tem sua própria janela na tela. Quando o usuário posiciona o mouse numa janela e faz uma opção, o sistema operacional notifica o programa específico e a operação se realiza.

Por exemplo: se o usuário muda o cursor para um canto de uma janela e seleciona a opção que transfere essa janela para nova posição, as rotinas do sistema operacional realizam a tarefa. E, se necessário, informam o programa sobre as mudanças, para que ele possa retificar apropriadamente sua tela. Se o usuário leva o mouse para uma janela diferente, o programa original torna-se inoperante e o sistema operacional começa a trabalhar com o novo programa. Portanto, mudar de aplicação é tão simples quanto mover o mouse.

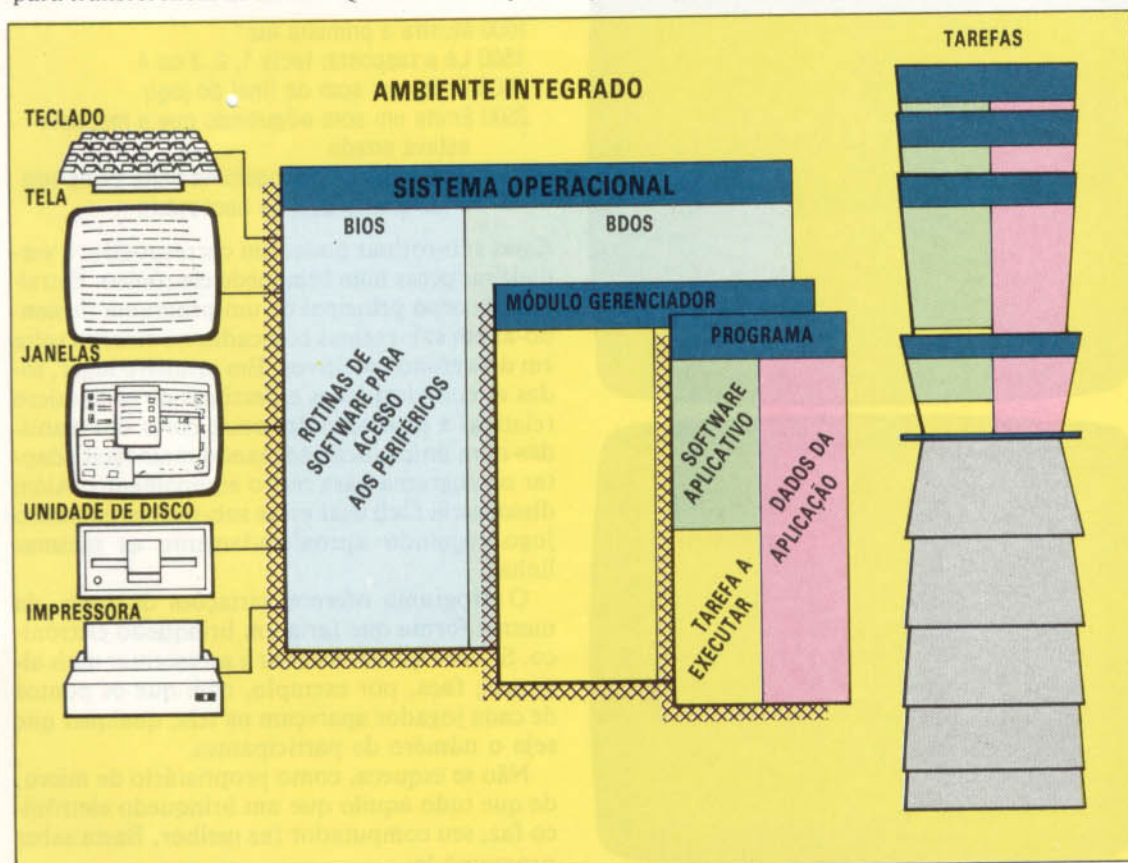
Todos os programas que estiverem na tela ao mesmo tempo encontram-se na memória e disponíveis para uso. Para facilitar o processo, muitos sistemas têm memórias maciças — por exemplo: 1 Megabyte no Lisa e 512 Kbytes no Macintosh. Mesmo assim, o sistema operacional precisa ocasionalmente armazenar e acessar em disco informações e partes de programas, a fim de acomodar tudo. Para que o sistema se torne razoavelmente rápido, é necessário utilizar um disco rígido.

Os dados são facilmente permutados entre os programas, porque o sistema operacional dispõe de uma série incorporada de formatos e rotinas para transferência de dados. Quando se "exporta" uma informação de certo programa e se pede a outro que a "importe", o sistema operacional suspende o primeiro programa e aciona o segundo. Então, pede ao aplicativo corrente que leia e processe a informação que está vindo de outro programa.

Esses caminhos se estabelecem automaticamente, para que, quando você alterar dados numa planilha, por exemplo, um gráfico da mesma planilha também seja alterado.

Os dois programas não rodam concomitantemente; o sistema operacional apenas faz as trocas entre eles, sempre que necessário.

O Lisa da Apple apresenta um conceito mais sofisticado. Nessa máquina, a informação pode ser "recortada" de uma janela com qualquer programa e "colada" em outra. Os dados são transferidos junto com suas informações de formatação, de modo que um gráfico produzido num gerador de gráficos seja transferido para qualquer outro programa sob a forma de gráfico mesmo.

Essa é a forma mais sensata de integrar softwares. Ela permite unir e combinar quaisquer programas no sistema, mover-se de um programa para outro e transferir dados entre eles. A desvantagem é que esse sistema requer um hardware sofisticado — e muito caro. Além disso, há poucos programas disponíveis para a integração. Qualquer grande inovação tecnológica demora para se tornar comum. A tecnologia do mouse e das janelas, por exemplo, só apareceu no mercado dez anos depois de desenvolvida pelas equipes de pesquisa da Xerox.

SISTEMAS

BIOS (BASIC Input/Output System): Elo de ligação lógica entre o hardware e o software aplicativo, que controla todas as operações de entrada e saída. Residente em ROM.

BDOS (BASIC Disk Operating System): Núcleo do sistema operacional, que detém as funções de controle do disco.

Operações combinadas
Num sistema integrado, o sistema operacional é reforçado pela adição de um módulo de gerenciamento, que trata todos os programas e dados correntes como tarefas a serem programadas e executadas. O módulo interage com o sistema operacional como simples software de suporte de sistema. Tira e põe tarefas na memória principal e nos discos, de acordo com os pedidos do usuário e com as necessidades da tarefa atual. Está equipado para passar informações de uma para outra aplicação em formato padronizado e permite assim a transferência de dados entre as tarefas. O gerenciador é, ele mesmo, uma tarefa de alta prioridade.

CORES E NÚMEROS

Tal como alguns brinquedos eletrônicos, o jogo aqui examinado baseia-se na antiga brincadeira Seu Mestre Mandou. O programa usa cores e sons para entreter por meio de um estimulante teste de memória.

No jogo conhecido como Seu Mestre Mandou, o líder dá instruções do tipo "Seu mestre mandou pôr as mãos na cabeça" ou "Seu mestre mandou ficar em pé" etc. Os jogadores serão desclassificados se obedecerem a uma instrução que não comece por "Seu mestre mandou".

Essa brincadeira de salão transformou-se num jogo eletrônico (no Brasil, o Genius, da Estrela), quando se desenvolveu uma série de brinquedos que utilizam microprocessadores para controlar um conjunto de botões, luzes e campainhas. Na versão computadorizada, a máquina é o líder.

O brinquedo eletrônico fornece uma seqüência de sons e cores e os jogadores devem repeti-la, pressionando as teclas apropriadas. A máquina então acrescenta outro som à seqüência. Esses jogos tornaram-se tão populares que muitos imaginam o Seu Mestre Mandou como sendo exclusivamente eletrônico.

Se você possui um micro, a idéia de um brinquedo eletrônico que só joga Seu Mestre Mandou deve parecer muito estranha. Um brinquedo pode ser portátil, durável e fácil de usar, mas mesmo os melhores jogos logo se tornam tediosos. Já a capacidade do computador de executar centenas de programas diferentes assegura que ele nunca ficará desinteressante.

O programa que aqui listamos é o Repita!, que executa um jogo baseado em quatro cores, cada qual correspondendo a um som e a uma tecla (1, 2, 3 ou 4). Uma luz acende e ele pede para ser imitado. Se você acertar, o programa exibirá duas luzes, e assim por diante.

Há dois modos pelos quais você perde esse jogo: se demorar demais para pressionar a tecla correta ou se pressionar a tecla errada três vezes numa partida. Para tornar o desafio mais difícil, essa versão também vai ficando cada vez mais rápida, embora não haja necessidade de se repetir a seqüência na velocidade em que ela é executada pelo computador. O jogo tem um máximo de até cinqüenta luzes numa seqüência, mas é pouco provável que você alcance esse limite. A maior parte dos jogadores mal consegue chegar a quinze luzes.

Vale a pena estudar o modo como o programa é estruturado. Todos os comandos de cor e som estão agrupados em sub-rotinas no final do programa deste modo:

1000 Mostra a primeira luz
1500 Lê a resposta: tecla 1, 2, 3 ou 4
2000 Emite um som de final do jogo
2500 Emite um som advertindo que a resposta estava errada
6000 Mostra uma mensagem na linha 21 da tela e faz uma pausa, se necessário

Essas sub-rotinas poderiam corresponder a verdadeiras peças num brinquedo eletrônico. Extraí-las do corpo principal de um programa (isolando-as em sub-rotinas colocadas no final) resulta em dois efeitos positivos. Em primeiro lugar, todas as complexidades específicas de cada micro relativas à produção de sons e cores são mantidas num único local, tornando mais fácil adaptar o programa para outro equipamento. Além disso, seria fácil usar essas sub-rotinas em outro jogo seguindo aproximadamente as mesmas linhas.

O programa oferece variações do jogo, da mesma forma que faria um brinquedo eletrônico. Se você quiser inventar e acrescentar mais algumas, faça, por exemplo, com que os pontos de cada jogador apareçam na tela, qualquer que seja o número de participantes.

Não se esqueça, como proprietário de micro, de que tudo aquilo que um brinquedo eletrônico faz, seu computador faz melhor. Basta saber programá-lo.

Memória prodigiosa
As combinações de cores e sons geradas pela versão para micros do jogo Seu Mestre Mandou são praticamente inesgotáveis e ilustram a grande vantagem dos computadores sobre nós: eles nunca esquecem nada, ao contrário da limitada memória humana.

Repita!

```
10   REM ********************************
20   REM **                            **
30   REM **          REPITA            **
40   REM **                            **
50   REM **       PARA O APPLE         **
60   REM **                            **
70   REM ********************************
80   REM
90   RAM **   DADOS PARA GERAR SONS    **
100  DATA 172,1,3,174,1,3,160,4,32,168,2
     52,173,48,192,232,208,253,136,208,2
     39,206,0,3,208,231,96
110  FOR Z=770 TO 795:READ BY:POKE Z,BY:
     NEXT:POKE 799,0:POKE 800,0
120  REM **      INICIALIZA O JOGO     **
130  DIM C(4),P(4),A(50):H=0:N=0:W=3
140  P(1)=5:P(2)=8:P(3)=12:P(4)=15
150  C(1)=1:C(2)=2:C(3)=4:C(4)=6
160  B$=CHR$(127):REM ** BLOCO CHEIO  **
170  HOME:PRINT TAB(10);"REPITA !":PRINT
     :PRINT
180  IF N>0 THEN PRINT:PRINT"VOCE CONSEG
     UIU "N" SEQUENCIAS"
190  IF N>H THEN PRINT:PRINT"ESTE E O NO
     VO RECORDE !":H=N
200  IF H>0 THEN PRINT:PRINT"O RECORDE A
     TUAL E "H" SEQUENCIAS"
210  PRINT:PRINT"TENTE REPETIR AS SERIES
     DE LUZ E SOM DO COMPUTADOR USANDO A
     S TECLAS DE 1 A 4"
220  PRINT:PRINT"USE J PARA JOGAR OU P P
     ARA PARAR"
230  GET A$:IF A$="" THEN 230
240  IF A$="P" THEN HOME:END
250  IF A$<>"J" THEN 230
260  HOME:PRINT TAB(10);"REPITA !"
270  FOR A=1 TO 4:GOSUB 1000:NEXT A
280  N=0:M=0
290  N=N+1
300  A(N)=INT(RND(1)*4)+1
310  IF M=W THEN GOSUB 2000:W$=STR$(W):M
     $="*"+W$+" RESPOSTAS ERRADAS":GOSUB
     6000:GOTO170
320  M$="*ATENCAO !":GOSUB 6000
330  FOR I=1 TO N
340  A=A(I):GOSUB 1000
350  FOR J=1 TO 100/N:NEXT J:NEXT I
360  M$="REPITA A SEQUENCIA":GOSUB 6000
370  I=1
380  GOSUB 1500:IF T=0 THEN GOSUB 2000:M
     $="*MUITO DEVAGAR":GOSUB 6000:GOTO
     170
390  IF A<>A(I) THEN M=M+1:GOSUB 2500:GO
     TO 310
400  I=I+1:IF I<=N THEN 380
410  IF N>50 THEN M$="GANHOU COM 50 SEQU
     ENCIAS !!!":GOSUB 6000:GOTO 170
420  M$="*PREPARE-SE PARA A PROXIMA":GOS
     UB 6000
430  GOTO 290
1000 REM ********************************
1010 REM **  ROTINA GERADORA DE LUZ    **
1020 REM ********************************
1030 COLOR=C(A)
1040 P=(A-1)*8+2
1050 FOR L=10 TO 14
1060 VTAB L:HTAB P:PRINT;
1070 IF L=12 THEN PRINT B$;B$;A;B$;B$
1080 IF L<>12 THEN PRINT B$;B$;B$;B$;B$;
1090 NEXT L
1100 REM **  ROTINA GERADORA DE SOM    **
1110 POKE 768,2:POKE 769,A*50:CALL 770
1120 VTAB 10:HTAB P:PRINT"     ".
1130 VTAB11:HTABP:PRINT" ";B$;B$;B$;" ";
1140 VTAB12:HTABP:PRINT" ";B$;A;B$;" ";
1150 VTAB13:HTABP:PRINT" ";B$;B$;B$;" ";
1160 VTAB 14:HTAB P:PRINT"     "
1170 RETURN
1500 REM ********************************
1510 REM **    VERIFICA    TECLADO     **
1520 REM ********************************
1530 T=250
1540 GET A$:IFA$="" THEN 1540
1550 A=VAL(A$):IFA<0 OR A>4 THEN 1540
1560 IF A=0 THEN RETURN
1570 GOSUB 1000
1580 RETURN
2000 REM
2010 REM **   RUIDO  DO FIM DO JOGO    **
2020 REM
2030 FOR Z=1 TO 200 STEP 20:POKE 768,1:P
     OKE 769,Z:CALL 770:NEXT Z:RETURN
2500 REM
2510 REM **       AVISO DE ERRO        **
2520 REM
2530 POKE 768,1:POKE 769,0:CALL 770
6000 REM ********************************
6010 REM **       EXIBE MENSAGEM       **
6020 REM ********************************
6030 VTAB 21:PRINT M$
6040 FOR Z=1 TO 400:NEXT Z
6050 S$=" ":FOR Z=1 TO 6:S$=S$+S$:NEXT Z
6060 VTAB 21:PRINT S$
6070 RETURN
```

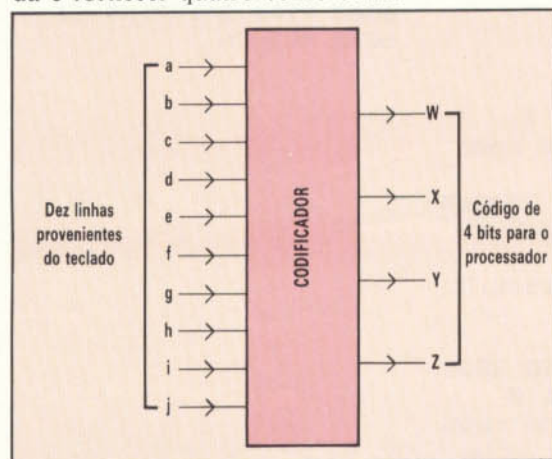
MUDANÇAS DE CÓDIGOS

A CPU dos micros desempenha sua função pelo envio de instruções, sob forma de impulsos elétricos, para dispositivos internos ou periféricos, que as traduzem em sinais elétricos e vice-versa.

O processador envia instruções para os dispositivos internos (como o acumulador) ou para os periféricos do equipamento (como a impressora). Muitas vezes, reduz-se o número de linhas usado por um periférico a fim de fornecer uma entrada simplificada para o processador — é a codificação.

Um exemplo de codificador seria o circuito usado em conjunção com o teclado. Ele pode ter 64 linhas de saída, e uma delas produz sinal quando se pressiona a tecla correspondente. Como uma tecla é acionada a qualquer momento, cada um dos 64 sinais possíveis de saída pode ser codificado como um número binário de 6 bits. Isso significa que seis linhas bastam para transportar ao processador a indicação de que tecla foi pressionada. O dispositivo que converte 64 linhas em seis é o codificador.

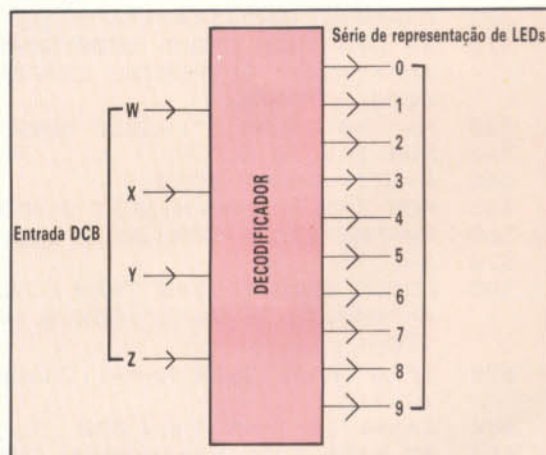
Na prática, mais duas linhas são acrescentadas — uma para verificar a paridade e outra para indicar quando uma tecla [Shift] ou [Control] foi usada juntamente com outra.

Para demonstrar esse princípio, consideremos um teclado muito simples, que possua apenas dez teclas. Isso possibilita digitar um número de 0 a 9. Um código binário de 3 bits permite apenas oito combinações; assim, temos de desenhar um codificador para admitir dez linhas como entrada e fornecer quatro como saída.

Como só uma das dez linhas pode ser ativada em dado momento, a tabela de validação para o codificador será:

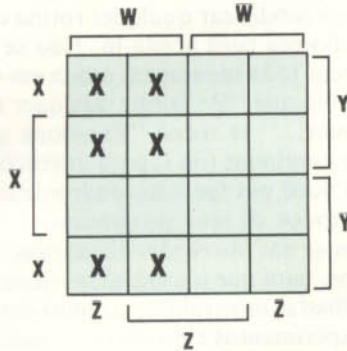
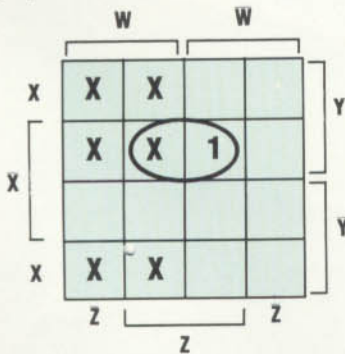
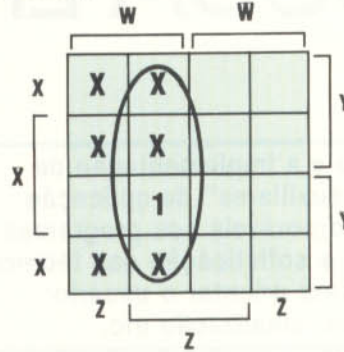
Decimal	Entradas										Saídas			
	a	b	c	d	e	f	g	h	i	j	W	X	Y	Z
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	1	0	0	0	0	0	0	0	0	0	1	0
3	0	0	0	1	0	0	0	0	0	0	0	0	1	1
4	0	0	0	0	1	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	1	0	0	0	0	0	1	0	1
6	0	0	0	0	0	0	1	0	0	0	0	1	1	0
7	0	0	0	0	0	0	0	1	0	0	0	1	1	1
8	0	0	0	0	0	0	0	0	1	0	1	0	0	0
9	0	0	0	0	0	0	0	0	0	1	1	0	0	1

A decodificação é o inverso da codificação. Em lugar de admitir grande número de linhas de entrada para produzir poucas de saída, o decodificador admite pequeno número de entradas (em geral sob a forma de códigos binários provenientes do processador). A partir delas, seleciona uma entre as várias linhas de saída que controlam a atividade de um periférico (impressora ou traçador de gráficos tipo x-y, por exemplo).

Codificadores e decodificadores também são usados para controlar os movimentos da cabeça de disco e para selecionar canais de saída por meio dos números dos periféricos.

Vejamos como um decodificador simples pode ser desenhado com o uso das portas E, OU e NÃO, examinando o problema seguinte. É necessário um decodificador para converter decimais codificados em binários (DCB) numa forma que ligue uma das dez luzes (LED) correspondentes ao valor decimal do código. Aqui estamos lidando com um circuito que produz o inverso do exemplo do codificador.

Os DCBs são representações de 4 bits dos dígitos decimais de 0 a 9; assim, o decodificador tem quatro linhas de entrada. Como qualquer combinação das quatro linhas pode ser ativada, há dezesseis entradas possíveis. Preservamos as dez primeiras combinações desse tipo e aplicamos

condições (X) de entrada não válida às outras. A tabela 1 é sua tabela de validação. A expressão booleana para cada saída parece exigir quatro termos (W, X, Y e Z), como na tabela 2.

Como as condições de entrada não válida são as mesmas para cada uma das dez saídas, podemos unificar os termos para simplificá-los. Abaixo, o mapa de Karnaugh, com quatro variáveis, contém as seis condições de entrada não válida assinaladas.

Se examinarmos uma das saídas por vez, verificaremos que ainda podem ser feitas simplificações. Tomemos, por exemplo, a saída para o dígito 3. Pela colocação da expressão booleana no mapa-k, verificamos que é possível desenhar um loop que a inclua.

Assim, a expressão para a saída pode ser simplificada para $\overline{X}.Y.Z$. Tomando a saída 9 como um segundo exemplo, temos:

Pode-se desenhar um loop que utilize três das condições de entrada não válida e que represente a expressão booleana W.Z. Muitos dos outros termos de entrada fazem a simplificação de forma semelhante. Talvez você queira verificar se os termos da saída simplificada são os apresentados na tabela 3.

A tarefa agora é construir o diagrama de circuitos com as dez expressões booleanas. Como se usa cada entrada tanto na forma normal como na negativa, fica mais fácil construir o circuito de oito linhas paralelas representando esses termos. Cada entrada é formada desviando-se dessas linhas e usando-se portas E. Na parte inferior da página apresentamos o diagrama de circuito decodificador completo.

Exercício 6

1) Projete um codificador de três entradas que crie uma saída de 1 para entradas de 011, 101, 110 ou 111. A saída deverá ser 0 para outras combinações de entrada.

a) Desenhe a tabela de validação para o codificador.

b) Crie uma expressão booleana para a saída e simplifique-a.

c) Desenhe o circuito codificador.

CIRCUITO DECODIFICADOR

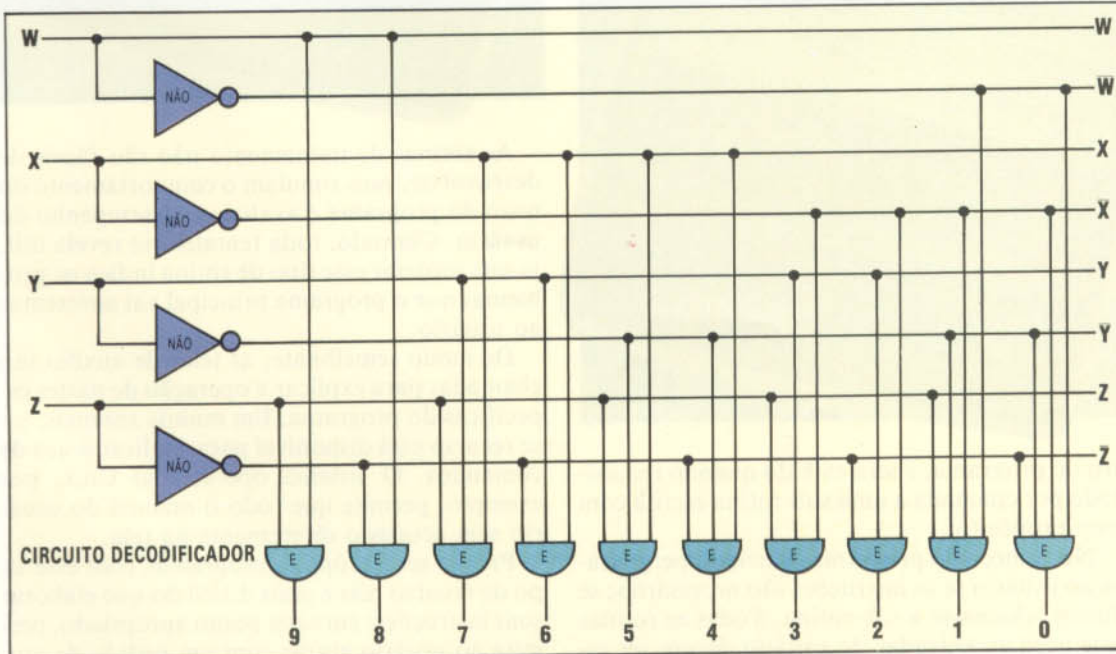
Tabela 1

Entradas				Dig. DCB
W	X	Y	Z	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Tabela 2

Dígito DCB	Expressão booleana
0	$\overline{W}.\overline{X}.\overline{Y}.\overline{Z}$
1	$\overline{W}.\overline{X}.\overline{Y}.Z$
2	$\overline{W}.\overline{X}.Y.\overline{Z}$
3	$\overline{W}.\overline{X}.Y.Z$
4	$\overline{W}.X.\overline{Y}.\overline{Z}$
5	$\overline{W}.X.\overline{Y}.Z$
6	$\overline{W}.X.Y.\overline{Z}$
7	$\overline{W}.X.Y.Z$
8	$W.\overline{X}.\overline{Y}.\overline{Z}$
9	$W.\overline{X}.\overline{Y}.Z$

Tabela 3

Dígito DCB	Expressão booleana
0	$\overline{W}.\overline{X}.\overline{Y}.\overline{Z}$
1	$\overline{W}.\overline{X}.\overline{Y}.Z$
2	$\overline{X}.Y.\overline{Z}$
3	$\overline{X}.Y.Z$
4	$X.\overline{Y}.\overline{Z}$
5	$X.\overline{Y}.Z$
6	$X.Y.\overline{Z}$
7	$X.Y.Z$
8	$W.\overline{Z}$
9	$W.Z$

AJUDA EM LINHA

O projeto e a implementação de rotinas "auxiliares" de aplicação geral incorporáveis aos programas mostram a sofisticação das técnicas usadas para orientar o usuário: instruções, sinalização etc.

Os chips de memória se tornam progressivamente mais baratos e contêm cada vez mais informações. A próxima geração de computadores domésticos disporá de muito mais memória do que requerem nossos mais ambiciosos programas. Assim haverá espaço suficiente para melhores instruções, mensagens de erro e auxílio on-line ao usuário.

Três tipos principais de apoio ao usuário podem ser supridos dentro de um programa: instruções, telas de auxílio e sinalização.

As instruções tomam duas formas: são fornecidas num único bloco no começo do programa ou apresentadas conforme requisitadas no decorrer dele (por exemplo, mensagens para a entrada de dados pelo usuário). Idealmente, ambas devem estar disponíveis.

Em sua forma mais simples, as instruções compõem uma ou várias telas de texto, em linguagem corrente, sobre como usar o programa. O texto pode ser armazenado em strings (variáveis alfanuméricas) ou nos comandos DATA, dentro do programa, e será exibido quando requisitado por chamada a uma sub-rotina escrita com esse propósito.

Auxílio permanente
O processador de texto WordStar da Micropro fornece um exemplo de grande aceitação de software guiado por comandos com auxílio on-line. O menu de ajuda que fica na tela pode ser abreviado ou eliminado pelo usuário, mas uma estrutura de telas de auxílio bem detalhadas está sempre disponível.

No começo do programa principal, pergunta-se ao usuário se as instruções são necessárias; se forem, chama-se a sub-rotina. Todas as rotinas que aceitam entradas do usuário devem ser escritas de forma que uma entrada específica (? ou I) chame a sub-rotina das instruções.

Convém padronizar o comando que mostra as instruções e modificar qualquer rotina de entrada da biblioteca para aceitá-lo. Não se esqueça de modificar toda mensagem usada em sua rotina, de forma que "Pressione qualquer tecla para continuar..." se torne "Pressione qualquer tecla para continuar (ou I, para instruções)". Isso dará a você um formato padronizado, utilizável em todos os seus programas.

Incluem-se nas instruções diagramas, detalhes e exemplos, para que o usuário pratique. Instruções detalhadas mostram-se comuns em programas de experimentos científicos — nos quais se pede ao usuário para executar uma tarefa específica, de determinado nível de habilidade, antes que lhe seja permitido entrar no programa principal.

As rotinas de treinamento não são fáceis de desenvolver, pois simulam o comportamento do resto do programa e avaliam o desempenho do usuário. Contudo, toda tentativa se revela útil, já que projetar esse tipo de rotina indica os problemas que o programa principal vai apresentar ao usuário.

De modo semelhante, as telas de auxílio são chamadas para explicar a operação de partes específicas do programa. Em muitos sistemas, esse recurso está disponível para explicar o uso de comandos. O sistema operacional Unix, por exemplo, permite que todo o manual do usuário seja acessado diretamente na tela.

Prover seus próprios programas com esse tipo de recurso não é mais difícil do que elaborar suas instruções: em cada ponto apropriado, permita ao usuário entrar com um pedido de aju-

da. Quando isso acontecer, o programa deve chamar a rotina de auxílio adequada.

Um programa complexo exige muitas telas de auxílio, sendo então necessário que se crie uma rotina geral com tal fim. No caso, o usuário fornece um número para identificar a tela solicitada. Em sistemas que disponham de disquetes, as telas de auxílio são armazenadas como arquivos separados. A rotina de auxílio acessa, então, o arquivo apropriado, a partir da entrada do usuário, e mostra seu texto na tela.

Quando longos, os textos de auxílio ou de instruções podem ocupar mais do que uma tela. Se for o caso, a rotina precisa ser projetada de maneira que o usuário possa acessar, de acordo com sua necessidade, telas mais à frente e mais atrás.

Você também deve assegurar-se de que o usuário possa abandonar a rotina em qualquer estágio e retornar ao ponto exato em que o programa principal foi deixado.

Cabe à rotina de auxílio fazer uma assinalação (flag) que informe à rotina de chamada que ela precisa voltar à última instrução antes do pedido de ajuda, e zerar, então, o flag.

Uma metáfora comum para a interação do usuário com um programa complexo é imaginar que ele navega por uma emaranhada rede de lógica. Principiantes no programa não entendem sua estrutura e podem ficar desnorteados. Assim sendo, a sinalização é necessária para orientar o usuário. Um menu é o exemplo mais claro: funciona como uma placa de trânsito que mostra as saídas possíveis a partir de um cruzamento.

Algumas instruções são mais importantes do que outras. Num sistema baseado em comandos, existem dezenas de comandos possíveis. No entanto, nem todos são relevantes, ou mesmo possíveis, estando-se em determinado ponto do programa. Se o número de opções for pequeno, convém mostrar uma ou duas linhas para explicar o que faz cada opção.

Algumas opções — como a de encerrar o programa — devem estar sempre disponíveis ou mesmo em exibição permanente. É conveniente também que os comandos para gravação, correção e outros, de aplicação específica, estejam constantemente disponíveis. Uma técnica comum consiste em programar tais comandos nas teclas de função e exibir uma mensagem de uma só linha mostrando a função de cada tecla. Indicar a possibilidade de uma saída para fora do programa dá segurança a quem o esteja utilizando pela primeira vez.

Boa administração

O Manager, sistema gerenciador do micro Apricot, guia o usuário na escolha dos programas utilitários apresentados em menus. O auxílio é uma opção permanente, e consiste numa explicação dos outros itens. Esse é um bom exemplo de software orientado por menus, com extensos arquivos de auxílio.

Alguns sistemas experimentais são capazes de monitorar o desempenho do usuário e ajustar o nível de auxílio fornecido de acordo com o desempenho. Se o programa pede o nome do usuário a cada vez que roda, pode-se manter um arquivo sempre atualizado dos usuários e seus níveis de habilidade.

Esses níveis são calculados a partir do número de vezes em que cada um deles executou o programa ou pelo maior número de pontos conseguidos, se o programa em questão for um jogo. À medida que aumenta o nível de habilidade, o tipo de ajuda e de sinalização muda, tornando-se mais breve e interferindo menos na execução do programa. Pode-se, também, pedir ao usuário que escolha o nível de auxílio, como no processador de texto WordStar.

A inclusão de instruções de auxílio pode ser um método valioso para melhorar o desempenho do programa. Uma vez projetada a rotina de auxílio, fica fácil modificá-la para que grave quais instruções foram requisitadas e com que freqüência. Isso dá clara indicação dos pontos problemáticos no programa.

OBJETIVO BRASILEIRO

Diversificação
Da linha de produção da Scopus, instalada na fábrica da foto, saem computadores tanto de uso pessoal como para aplicações profissionais, além de terminais de vídeo e de software especializado.

Pioneirismo
O primeiro computador projetado no Brasil foi o Patinho Feio, montado em 1972 pela equipe do Laboratório de Sistemas Digitais da USP, da qual participaram os fundadores da Scopus.

O desenvolvimento de tecnologia própria tem sido a principal preocupação da Scopus desde sua origem. Em 1983, ela entrou no promissor mercado dos microcomputadores de 16 bits.

Em 1975, Josef Manasterski, Célio Yoshiyuki e Edson Fregni — professores da Escola Politécnica da Universidade de São Paulo (USP) — fundaram a Scopus, que iniciou suas atividades como prestadora de serviços, tanto na manutenção de computadores como na pesquisa e projetos de informática. Mas logo ela passou a produzir, em pequena escala, equipamentos digitais para aplicações específicas.

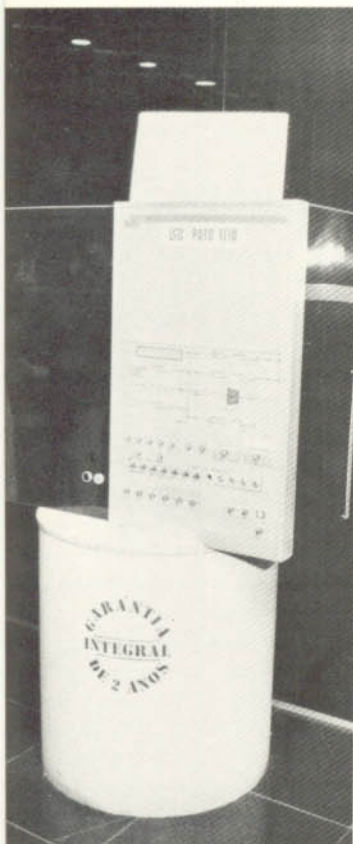
Experiência não faltava aos três professores: além de participarem dos projetos, desenvolvidos em 1971-72 na USP para nacionalizar a tecnologia dos computadores, um deles — Manasterski — foi, no período 1973-75, o coordenador da equipe que criou o primeiro computador de médio porte da Politécnica, o G-10.

No início de 1985, todos os equipamentos produzidos pela Scopus eram inteiramente projetados por seus engenheiros e analistas, dos quais noventa dedicados à pesquisa e ao desenvolvimento de hardware e software.

Essa trajetória, marcada pela independência tecnológica, teve início em 1976, com o lançamento de um terminal não inteligente para computadores de médio e grande porte. Ele serviu de base para a criação do terminal inteligente TVA 800, compatível com a família de computadores fabricados pela Burroughs.

Ate 1981, esse equipamento foi vendido apenas a fabricantes (nacionais e estrangeiros) de sistemas de computação. Com a decisão de investir em terminais de vídeo compatíveis com os equipamentos IBM e em microcomputadores, a Scopus passou a comercializar seus produtos diretamente com o usuário final.

A estratégia da Scopus para ampliar sua participação no mercado envolveu duas áreas prioritárias: o desenvolvimento de software para redes de processamento de dados — como o sistema de gerenciamento de dados SBD/TS — e a exploração do segmento dos microcomputadores profissionais.

A empresa paulista, que já vinha produzindo a linha Microscopus de 8 bits desde 1981, lançou em 1983 o Nexus 1600, acompanhando a tendência mundial que privilegia os micros de 16 bits e sua conexão aos grandes sistemas. O Nexus 1600 integra-se à maioria dos sistemas de grande porte e às redes de serviços públicos, como Renpac, Aruanda e Cirandão.

HOMENS E MÁQUINAS

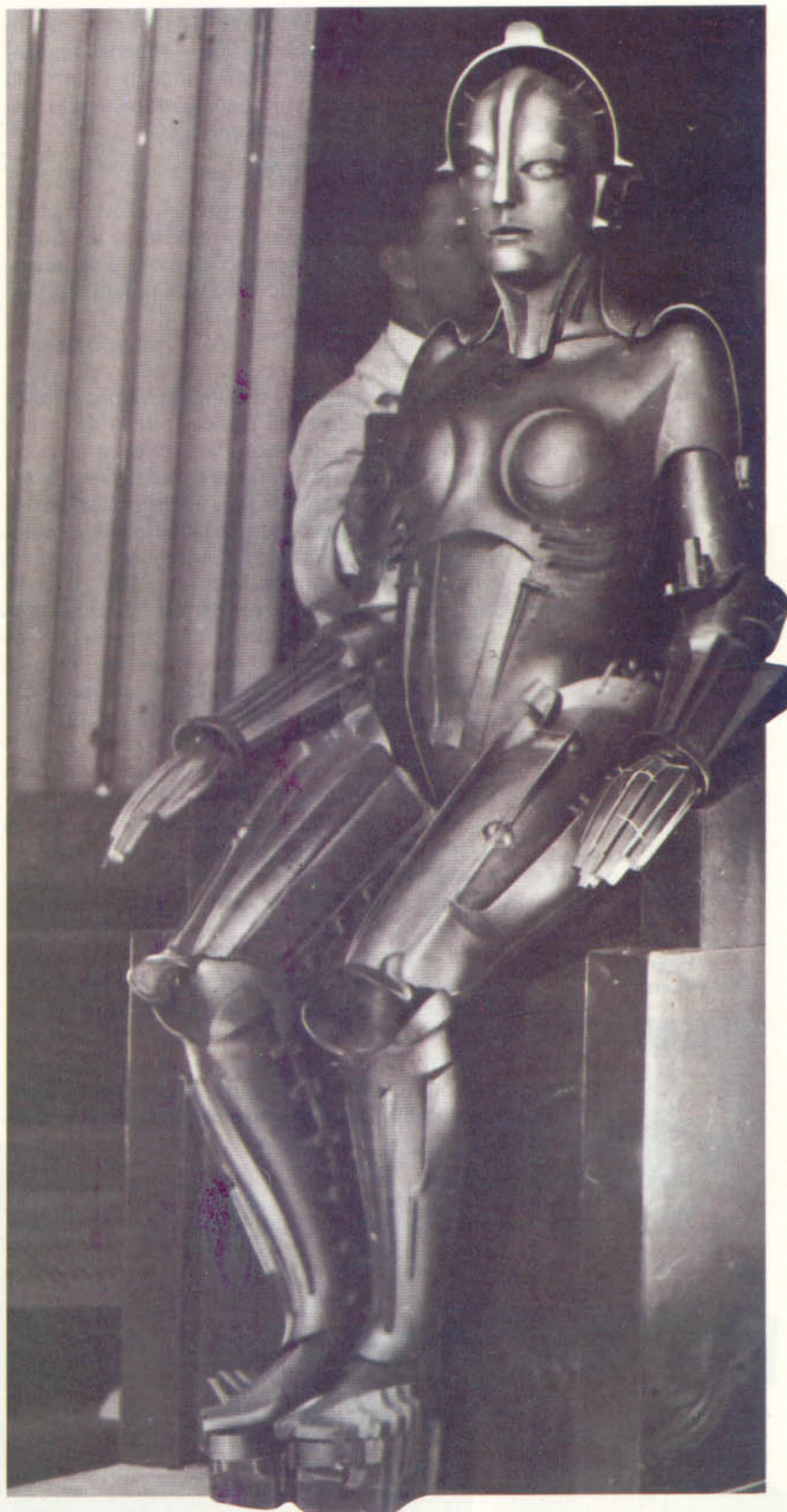
Máquinas que imitam o homem pela forma ou ação, os robôs constituem um tema sempre fascinante. Daí sua constante evolução, dos bonecos mecânicos do século XVIII aos robôs industriais de hoje.

Há séculos que as pessoas são atraídas pela idéia da existência de homens mecânicos. Filósofos, engenheiros, inventores, em várias épocas, pensaram na criação de mecanismos que imitassem o comportamento humano. Embora os robôs modernos tenham a aparência menos humana e sejam projetados para desempenhar uma série de ações específicas, geralmente nas indústrias, os bonecos mecânicos do passado eram projetados para ter a aparência mais viva possível e dar a impressão de que seriam capazes de praticar qualquer ação humana.

No entanto, o primeiro robô não se parecia com um homem: parecia um pato. Em 1738, Jacques de Vaucanson (1709-1782), um engenheiro francês, apresentou um pato mecânico à Academia Real de Ciências, em Paris. Seu invento batia as asas, grasnava e comia. Mais tarde, ainda no século XVIII, um inventor suíço, Pierre Jacquet-Droz (1721-1790), criou um conjunto de bonecos mecânicos capazes de variadas ações. Um escrevia, outro desenhava, outro tocava órgão. No final do século XIX já havia grande número desses brinquedos, todos baseados em mecanismos de relojoaria.

Os primeiros andróides

Nesse período, construíram-se numerosas figuras com aparência humana extraordinariamente verossímil, cujos movimentos nem sempre eram baseados em peças de relógio. Em 1893, George Moore fez um homem mecânico, movido a vapor, que produzia um efeito interessante: o vapor que saía de sua boca parecia a fumaça de um charuto que ele "fumava".

Tecnologias mais evoluídas, contudo, estimularam o processo de aperfeiçoamento dessas máquinas: desde simples brinquedos mecânicos capazes de andar até o clássico Elektro, andróide construído pela empresa americana Westinghouse. Com 2,15 m de altura, era capaz de dizer oitenta palavras, contar, andar, cumprimentar e reconhecer cores. Movido por onze motores elétricos, pesava 117 kg. Para controlar essa imensa figura, a Westinghouse desenvolveu um "cérebro" composto por 82 relés.

Mas cada um desses homens mecânicos tinha suas limitações. Além do valor como entretenimento, nenhum deles era hábil o bastante para ser considerado um robô ideal. Um homem mecânico que sabe desenhar não pode fazer as compras por você. E, mesmo sabendo andar por uma sala, seria incapaz de chegar a uma loja sem an-

O antepassado
Este robô foi o primeiro a aparecer no cinema: está no filme de Fritz Lang *Metropolis* (1926), que, em 1984, teve uma versão sonorizada e parcialmente colorida por computador.

tes se chocar com um poste. Todos eram máquinas, simplesmente máquinas — apenas realizavam uma série de ações que não requeriam tomada de decisões e não demonstravam nenhum sinal de inteligência.

Robôs na ficção

Mas, se os inventores e engenheiros esbarravam sempre em limitações, os escritores de ficção científica superavam todas com o uso da imaginação. Esse gênero de literatura desenvolveu-se bastante com a idéia do robô.

A própria palavra "robô" é produto de uma obra de ficção: foi criada em 1923 pelo autor tcheco Karel Capek (1890-1938). Na peça *Robôs universais de Rossum,* Capek tratava da invenção de homens mecânicos tão perfeitos que executavam todas as tarefas realizadas por um ser humano. No final, os robôs achavam que os humanos não serviam para nada, colocando-os em uma situação dramática. Em tcheco, a palavra *robota* significa "trabalhador". Assim, o título da peça deveria ser traduzido por "Trabalhadores universais de Rossum", mas a palavra "robô" surtiu inesperado efeito, transformando-se no termo padrão para qualquer ser mecânico com habilidades humanas (ou semelhantes).

As fantasias da ficção sobre criaturas construídas à imagem do homem encontram-se também no romance *Frankenstein* (1818), de Mary Shelley. Embora não fosse mecânico, o monstro criado por Victor Frankenstein era constituído por um conjunto de partes obtidas de outros seres humanos. As criaturas invasoras de *Guerra dos mundos* (1898), de H. G. Wells, eram robotizadas, ainda que parcialmente.

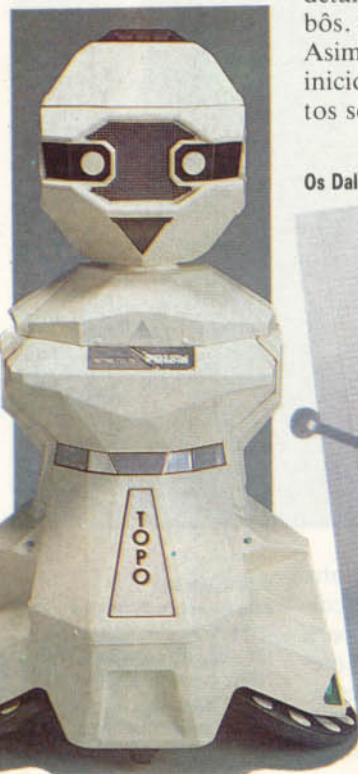
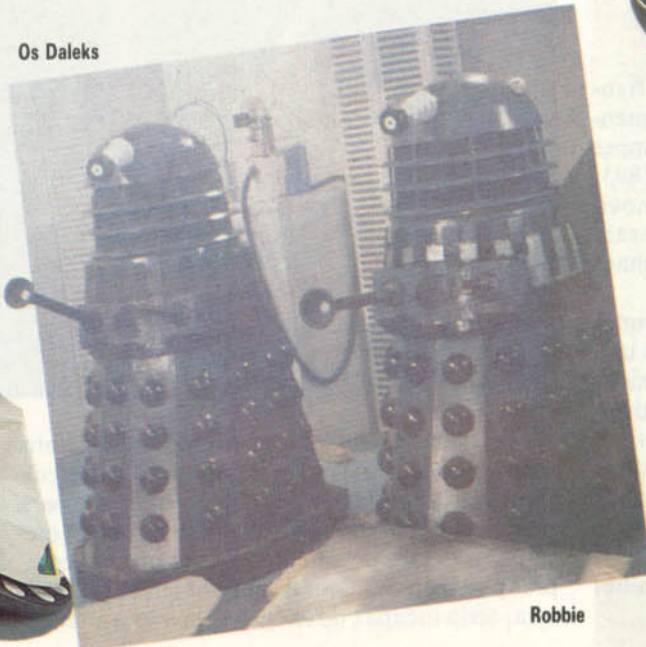
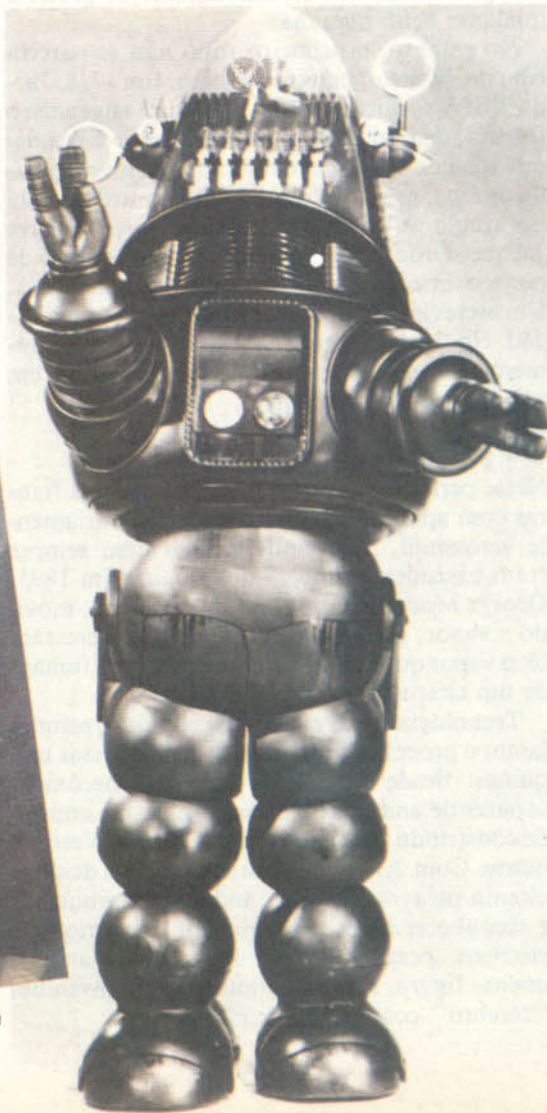
Os escritores do século XX têm explorado em detalhes um mundo imaginário habitado por robôs. A contribuição mais notável foi a de Isaac Asimov, famoso escritor de ficção científica, que iniciou sua carreira em 1940 escrevendo contos sobre robôs e seus problemas operacionais.

O mundo dos robôs de Asimov é tão bem estruturado que ele chegou a formular as três "leis da robótica" no "Manual de robótica (56.ª edição, 2058 d.C.)", do romance *Eu, robô.*

Os robôs também assinalam sua presença no cinema e na televisão. Por exemplo, nas séries de tevê sobre ciborgs *(O homem de 6 milhões de dólares* e *A mulher biônica)* e no filme *Guerra nas estrelas,* em que C3PO e R2D2 apresentam-se em igualdade com os astros humanos com quem contracenam.

Comparado a tais arrojos de fantasia, o emprego real dessas máquinas parece banalidade. Mas são os robôs industriais, encontrados nas linhas de montagem, que recebem a atenção dos homens de negócios: as estimativas para 1985 eram de 25.000 robôs em uso na indústria japonesa, 15.000 nos Estados Unidos e 8.000 na Alemanha Ocidental. No Brasil, eles começaram a ser usados na indústria automobilística, em 1982.

Para a maioria das pessoas, porém, os robôs industriais são desinteressantes. Uma máquina que passa o tempo todo soldando chassis de automóvel ou pintando carroçarias está longe da imagem criada pela ficção. Se o robô ideal virá um dia a ser desenvolvido ou não, é questão para discutir. E dificilmente se pode determinar se ele será projetado à imagem do homem.

Ficção e realidade
Os Daleks, certamente os robôs mais famosos do mundo, durante muitos anos foram heróis de um seriado de tevê europeu. Mas Robbie, de *Planeta proibido,* personificou a fantasia de um robô que tem até emoções. Todos muito diferentes de Topo *(abaixo),* robô de verdade para uso doméstico, produzido na Europa.

Os Daleks

Robbie

Os falsos homens

Pelo menos na ficção, os robôs receberam tantos nomes que se tornou necessário um glossário com os termos mais usados.

Andróide: Robô projetado de modo a parecer um ser humano, em todos os aspectos.

Antropomorfo: Significa, literalmente, "com forma humana". Assim, todo andróide é antropomorfo; mas muitos robôs só têm algumas partes com essa característica. Por exemplo, podem ter apenas braços antropomórficos.

Automação: Controle automático de qualquer processo de fabricação.

Autômato: Máquina com mecanismos ocultos que realiza uma série de operações predeterminadas. Os primeiros homens mecânicos eram autômatos. Mas a palavra tem significado mais profundo quando aplicada à teoria dos autômatos, um sistema analítico por meio do qual um dispositivo qualquer (um robô, um computador ou mesmo uma pessoa) pode ser estudado e descrito.

Cibernética: Estudo dos sistemas de controle e comunicações, tanto dos seres vivos como das máquinas. Criada por Norbert Wiener em 1947, baseia-se na possibilidade de ser usada para examinar sistemas biológicos como se estes fossem máquinas.

Cibert: Humanóide mecânico.

Ciborg: Do inglês cybernetic organism (organismo cibernético). Organismo no qual há partes mecânicas e partes biológicas.

Cibot: Robô que raciocina.

Colarinho metálico: Robôs industriais. Quem trabalha em escritório é designado como "colarinho branco"; se trabalha em produção, como "colarinho azul". Por analogia, o robô acabou sendo o "colarinho metálico".

Dróide: Robô que obedece às leis da robótica, de Asimov.

End effector: Termo técnico inglês usado para designar a mão de um robô.

Homúnculo: Homem pequenino ou boneco.

Manipulador: Mão de robô.

Robô: Máquina capaz de executar algumas funções humanas, embora não precise ter semelhança física com o homem.

Robótica: Ciência que estuda os robôs.

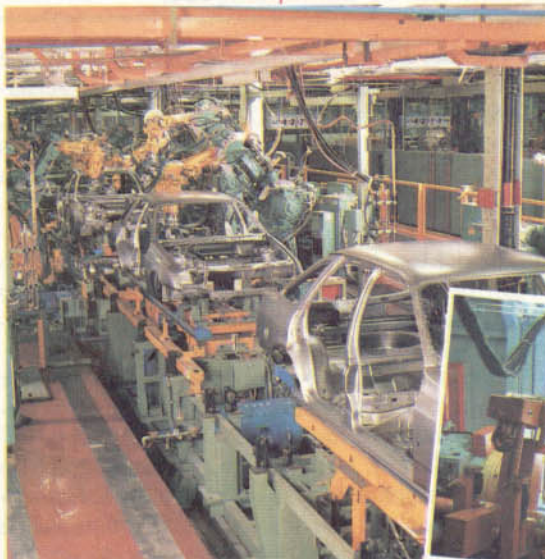
As leis da robótica

1. Um robô não pode ferir um ser humano nem permitir, por omissão, que um ser humano seja ferido.

2. Um robô deve obedecer às ordens dadas pelos seres humanos, exceto quando conflitarem com a primeira lei.

3. Um robô deve proteger sua existência enquanto isso não conflitar com as leis anteriores.

Do livro *Eu, robô*, de Isaac Asimov.

As regras
Quando os robôs forem capazes de distinguir um ser humano de uma pedra, as leis de Asimov *(acima)* poderão perfeitamente formar a diretriz de seu comportamento.

Ford: linha de montagem

Fiat: soldagem de componentes

Linha de montagem
Atualmente, os robôs são empregados sobretudo na indústria. Fazem o trabalho com mais rapidez e precisão que o homem e constituem ferramentas importantes em ambientes insalubres como as seções de soldagem e pintura das indústrias automobilísticas. Em geral, o robô todo é apenas um ou dois braços mecânicos.

O que é um robô?

Há dezenas de definições, mas no mundo inteiro o termo "robô" se tornou genérico para designar todas as máquinas que têm alguma semelhança com o homem e, também, máquinas industriais "inteligentes", dotadas de movimento e que podem ser programadas para inúmeras tarefas, como soldar, apertar porcas ou parafusos, pintar.

Teoricamente, um robô pode fazer tudo; as limitações ficam por conta de quem o projeta e da tecnologia disponível. Além de se movimentarem, os robôs "vêem", "ouvem", "falam". Certos dicionários já os definiram como máquinas capazes de executar algumas funções humanas (embora sem necessidade de se assemelharem ao homem), mas essa definição perdeu o sentido.

Sua forma e capacidade para desempenhar tarefas dependem apenas do que pretendemos e do que conseguimos que ele faça. Assim, pode ser que um robô de solda empregado numa indústria não se mova porque foi projetado para ficar apenas num lugar. Já um robô doméstico, embora capaz de fazer café, talvez não consiga subir as escadas e levar a bandeja até o quarto de seu dono.

DESEMPENHO VERSÁTIL

Embora tenha a aparência de simples máquina de escrever elétrica, a Brother EP-44 é muito versátil, e funciona como impressora, calculadora e terminal de comunicações.

A Brother EP-44 pesa pouco mais de 2 kg e é alimentada a pilhas, o que a torna uma verdadeira máquina portátil. Usando transformador, pode ser acionada por corrente elétrica. Além das teclas convencionais de máquina de escrever, a Brother EP-44 tem quatro outras para o processamento de texto e sete para a função de calculadora. Mas a característica mais notável é o visor de cristal líquido (LCD). Esse dispositivo possibilita ao usuário visualizar e corrigir o texto na tela antes de imprimi-lo.

Para usar a EP-44 como máquina de escrever comum, basta colocar o Print Mode Selector (seletor de modo de impressão) em Direct Print (impressão direta). Nesse modo, o texto aparece no LCD. Margeamento, tabulação e espacejamento de linhas são fixados como nas máquinas de escrever convencionais. No entanto, há um sistema incomum de alimentação de papel — em vez de girar manualmente o rolo, o usuário pressiona um botão a fim de mover o papel para cima ou para baixo.

A EP-44 utiliza uma cabeça de impressão térmica para queimar uma série de pontos no papel e assim formar o caractere desejado. A maioria das impressoras térmicas tem matriz de 9 x 7 pontos para formar os caracteres, mas a matriz de 24 x 18 da EP-44 fornece melhor qualidade de impressão.

A maior desvantagem do sistema térmico de impressão está na baixa velocidade da impressão — nesse caso, menos de dezesseis caracteres por segundo. Quando a Brother é usada como máquina de escrever, isso não chega a ser um inconveniente, mas, como impressora acoplada a um micro, a diferença é considerável: uma boa impressora matricial é dez vezes mais veloz que a Brother. O outro problema com esse sistema está na necessidade do papel térmico especial, muito caro.

Se o seletor de modo de impressão é posicionado em Correction Print (impressão com correção), o LCD mostra toda a sua utilidade. Nesse modo, o texto aparece no visor e há um intervalo antes da impressão: o texto registrado no papel fica quinze caracteres atrás da letra que está

O teclado
Há duas teclas para maiúsculas com trava, uma tecla para símbolos e outra para função. As teclas de caracteres são multifuncionais, produzindo acentos e uma série de caracteres em línguas estrangeiras. Teclas de função controlam os modos de impressão, de cálculo e de memória, o ajuste das margens, as tabulações e o movimento do papel.

sendo digitada. Esse recurso permite que você altere qualquer um dos quinze últimos caracteres antes que sejam impressos — o que será útil se sua datilografia não for muito boa e constitui grande vantagem com relação aos líquidos corretores. A correção é feita no visor pelo uso das duas teclas de movimentação do cursor para escolher o caractere a ser alterado.

Há também o modo Line-by-Line (linha por linha), no qual a EP-44 espera até que a tecla [Return] seja pressionada, antes de imprimir uma linha inteira. Isso dá a chance de se corrigir toda uma linha de texto em vez de apenas os quinze últimos caracteres. Nesse modo, o LCD age como uma janela que se desloca ao longo da linha de texto por meio das teclas do cursor.

A EP-44 possui o modo Auto Return (retorno automático), no qual uma nova linha será iniciada se a barra de espaço for pressionada a seis toques (ou menos) do final da linha.

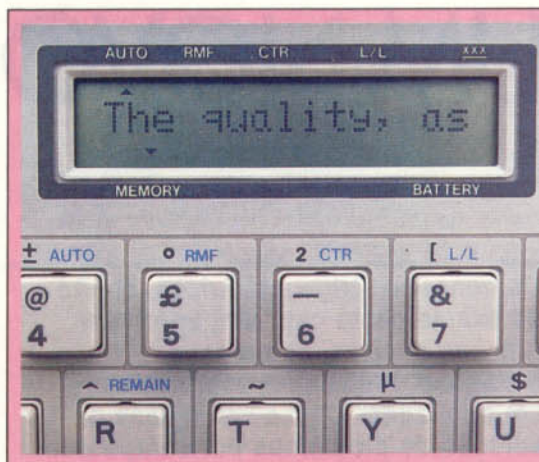
A calculadora incorporada na EP-44 possibilita a execução de contas no visor, sem prejudicar a impressão.

As diferenças

Todos esses recursos são encontrados em muitas das modernas máquinas de escrever elétricas. Mas a EP-44 pode também ser usada como uma processadora de texto simples. Para isso, dispõe de 3,5 Kbytes de memória interna para armazenar texto. Essa capacidade de memória parece pequena quando comparada à disponível na maioria dos micros, mas é suficiente para produzir uma carta de três páginas. Os comandos necessários para o processamento de texto são acessados pelo uso das teclas normais de máquina de escrever em conjunção com a tecla [Code]. Assim, para entrar com texto na memória, basta pressionar a [Code] e a letra N — que, no caso, significa "novo texto". A máquina aceita o texto tanto no modo direto como na impressão com correção.

Armazenado o texto na memória, usam-se o visor e as teclas do cursor para corrigi-lo. As teclas do cursor permitem mover o foco do LCD para cima, para baixo, para a esquerda e para a direita; assim, todo o texto pode ser visualizado e caracteres são inseridos, eliminados ou substituídos. A única restrição: não se deslocam palavras inteiras para a linha de baixo quando a linha que está sendo corrigida é muito longa — o que significa que você não pode reformatar toda a diagramação de um documento.

A grande vantagem de armazenar textos na memória está na possibilidade de modificá-los sem necessidade de redatilografar todo o documento. Se você quiser enviar a mesma carta a várias pessoas, precisará apenas modificar o texto na memória, inserindo o nome e o endereço desejados, e pressionar a tecla [Code] juntamente com a letra P (de Print, imprimir).

O texto permanece na memória da EP-44 mesmo depois de desligada, o que permite entrar com ele e corrigi-lo depois.

LCD

O visor de cristal líquido da EP-44 mostra quinze caracteres, cinco indicadores de modo e um aviso de descarga das pilhas. O texto pode ser editado na tela, permitindo processamento. A luminosidade dos caracteres no visor ajusta-se conforme as condições de iluminação do ambiente.

Uma chave marcada com a inscrição "Terminal" transforma a Brother em impressora de computador. Mas alguma experimentação é necessária para ajustar a velocidade de transmissão correta, o comprimento das palavras e outros aspectos relativos a seu micro.

É fácil compatibilizar a EP-44 com a interface serial do microcomputador: o visor mostra as várias velocidades de transmissão e outros parâmetros e você pressiona a chave Mode quando o parâmetro for adequado para sua máquina. Essa configuração mantém-se mesmo depois que se desliga a EP-44.

Versatilidade compactada

A Brother não pode lidar com formulários contínuos, e as folhas de papel devem ser colocadas manualmente, uma a uma. Mas a qualidade da impressão é muito melhor que a de uma impressora matricial; assim, a EP-44 é ideal para a produção de cartas e documentos curtos.

A Brother tanto envia como recebe dados. Conectada a um modem, permite que você se comunique por telefone com outros usuários, ou com máquinas maiores, que empregam padrões compatíveis de comunicação.

A Brother EP-44 destaca-se pela versatilidade e por reunir tantas funções diferentes em sua estrutura compacta. No mercado internacional, entra na avaliação de qualquer usuário de micro que esteja procurando uma impressora utilizável também como máquina de escrever.

Print-Out

Any smooth-finish typing paper can be used with the ribbon in place, but the paper-feed cannot accept carbons. The print quality is good on both types of paper.
Text can be
Centred & Underlined
foreign (Æø¿¢µ¶), and sub- or super-scripted ($X^2=N_2+T_0$).

Melhor de dois
Nesta foto são mostrados os vários formatos de impressão oferecidos pela EP-44, que combina os recursos de um micro com os de uma máquina de escrever.

UM NOVO ÂNGULO

Os conceitos trigonométricos, objeto de exame deste capítulo, envolvem funções básicas, como seno e co-seno, facilmente codificadas em linguagem BASIC e vantajosas na solução de problemas por computador.

De que tipo de matemática os programadores realmente precisam? Isso depende do programa que estão desenvolvendo. As diferentes versões da linguagem BASIC incorporam os inúmeros comandos e funções para lidar com desenhos: PLOT, CIRCLE, FILL, LINE, COLOR, INK, PAPER etc. Assim, a rotação e a translação de figuras simples na tela não apresentam grandes problemas, uma vez que alguns comandos já incluem os cálculos matemáticos necessários à realização desses movimentos. Há ocasiões, no entanto, em que são necessárias funções trigonométricas como SIN (seno), COS (co-seno) e TAN (tangente), o que também não constitui dificuldade porque a linguagem BASIC possui um bom conjunto delas. Se esses termos da trigonometria nada significam para você, não se preocupe — todos serão logo explicados.

Os estudantes de matemática do 2.° grau costumam perguntar sobre a relevância prática da trigonometria. Sob certo aspecto, a trigonometria é considerada uma ligação entre a geometria euclidiana, que lida com pontos, linhas e curvas, e a álgebra, que possibilita chegar a soluções matemáticas pela manipulação de variáveis com valores desconhecidos. A parábola, uma figura geométrica plana, serve como exemplo para comparar as soluções algébricas e geométricas. Suas propriedades podem ser observadas traçando-se a curva com o uso de papel quadriculado, transferidor, régua e lápis. No entanto, muito mais útil é perceber que a curva

2.° QUADRANTE

cos NÔR = valor de x negativo / hipotenusa

sen NÔR = valor de y positivo / hipotenusa

1.° QUADRANTE

sen NÔQ = valor de y positivo / hipotenusa

cos NÔQ = valor de x positivo / hipotenusa

eixo y — (+x, +y) Q — R (−x, +y) — P (a,p) — HIPOTENUSA — CATETO OPOSTO — TRIGONOMÉTRICA — θ TETA — ÂNGULO RETO — N eixo x — ORIGEM O — CATETO ADJACENTE — A (a,o) — CIRCUNFERÊNCIA — S (−x, −y) — T (+x, −y)

sen NÔS = valor de y negativo / hipotenusa

cos NÔS = valor de x negativo / hipotenusa

3.° QUADRANTE

sen NÔT = valor de y negativo / hipotenusa

cos NÔT = valor de x positivo / hipotenusa

4.° QUADRANTE

Sinais das funções

Podemos compreender as funções seno e co-seno desenhando uma série de triângulos retângulos dentro de uma circunferência. O lado da base é chamado cateto adjacente e o lado vertical é o cateto oposto. O seno de um ângulo é o comprimento do cateto oposto dividido pelo comprimento da hipotenusa (o lado oposto do ângulo reto). Para ângulos maiores que 90°, o cateto adjacente se torna negativo, o que resulta em valores negativos do co-seno.

pode ser representada pela fórmula algébrica $y = x^2$.

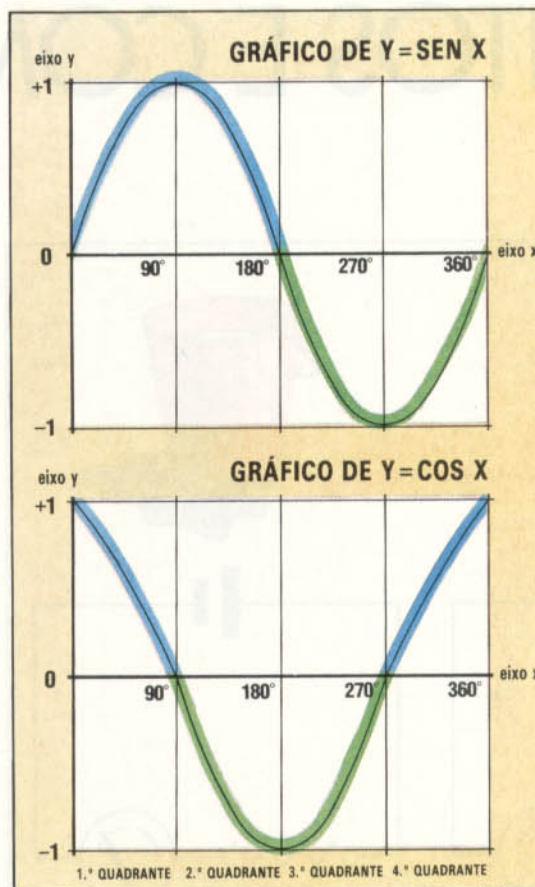
Essa fórmula permite calcular os valores para qualquer ponto da curva sem que se tenha de desenhá-la. Os problemas apresentados sob a forma algébrica prestam-se muito mais a soluções por computador do que os que requerem gráficos ou figuras; assim, torna-se evidente a vantagem das soluções trigonométricas.

O objetivo deste artigo é fazer uma rápida recapitulação dos conceitos trigonométricos, examinando suas funções básicas: seno e co-seno.

Razão entre lados

O seno e o co-seno de um ângulo ou de um arco representam a razão entre dois lados de um triângulo retângulo. Por se referirem a ângulos e arcos, essas funções estão diretamente relacionadas com a circunferência. Qualquer triângulo retângulo pode ser inscrito, ou seja, desenhado de forma que seus vértices estejam sobre a circunferência. A circunferência trigonométrica é também chamada unitária porque se atribui valor unitário a seu raio — a medida real não importa, são as proporções que interessam. A primeira ilustração mostra o segmento de reta ON, que descreve um ângulo θ, num movimento de rotação em torno do ponto O. A posição inicial de rotação, por convenção, é o eixo horizontal, no sentido anti-horário. O eixo horizontal é o eixo x ou eixo dos co-senos, e o ângulo de rotação costuma ser designado por letras do alfabeto grego. Traçando-se a partir de um ponto da circunferência uma linha perpendicular ao eixo x, obtém-se um triângulo retângulo.

O co-seno de θ é definido como a razão entre o cateto adjacente ao ângulo (lado OA, contido no eixo x) e a hipotenusa (OP, raio da circunferência unitária). Consideremos, por exemplo, um ângulo de 35° numa circunferência com raio de 15 mm: a medida do lado adjacente ao ângulo é de cerca de 12,28728 mm. Dividindo esse valor por 15, obtemos 0,819152, que é o valor do co-seno de 35°. Se você tiver uma calculadora científica com tecla [Cos] ou uma tabela de co-senos, tente entrar com [35 Cos]. Deverá obter o resultado 0,819152044. Essa razão é verdadeira para qualquer tamanho de circunferência unitária na qual o triângulo retângulo esteja inscrito. Tenha o raio do círculo 1 cm, 1 km ou um ano-luz, o lado do triângulo contido no eixo x sempre medirá cerca de 0,82 do valor do raio.

É possível também desenhar outros ângulos na circunferência unitária da ilustração. Pode-se ver que, para qualquer ângulo θ, o valor de cos θ (cos é a representação algébrica de co-seno) nunca será maior que 1 nem menor que 0. No entanto, para valores de θ maiores que 90°, o cos θ será negativo. Isso porque, à esquerda da origem (O), os valores das coordenadas x são negativos, e à direita, positivos. Assim, para ângulos θ maiores do que 90°, o cateto adjacente estará do lado negativo, o que implica valores negativos para cos θ. Pelo mesmo motivo,

Ondas

O desenho da senóide é produzido com os valores assumidos pela função seno numa circunferência completa. Observe que os eixos x e y aqui representados nada têm a ver com os eixos x e y que separam os quadrantes da circunferência trigonométrica. Ao longo do eixo x estão os ângulos de 0° a 360°, e no eixo y encontram-se os valores dos senos desses ângulos. Todos os valores do seno estão entre +1 e −1. As quatro seções do gráfico correspondem aos quadrantes; o seno é positivo (em azul) nos dois primeiros quadrantes e depois negativo (em verde). O gráfico do co-seno chega a resultados semelhantes: tem a mesma forma da senóide, apenas deslocado 90° ao longo do eixo x.

os ângulos entre 180° e 270° também terão co-senos negativos.

No entanto, entre 270° e 360° (o que corresponde a uma volta completa), o co-seno será novamente positivo.

Para maior facilidade de localização na circunferência trigonométrica, costuma-se dividi-la em quatro partes iguais — os quadrantes —, compreendendo cada uma ângulos de 90°. Assim, o 1.° quadrante abrange os ângulos de 0° a 90°; o 2.° vai de 90° a 180°; o 3.°, de 180° a 270°; e o 4.°, de 270° a 360°.

A função seno de um ângulo é muito semelhante ao co-seno, exceto que envolve valores no eixo y (o eixo vertical). Por uma simples translação, pode-se ver que os valores no eixo y coincidem com o comprimento do cateto oposto ao ângulo. Se θ for 0, o lado adjacente ao ângulo será igual ao comprimento da hipotenusa. A coordenada de P no eixo x será sempre 1 (porque $1/1 = 1$), mas a coordenada do eixo y será 0 (ou seja, o cateto oposto será 0). Para todos os valores de 0° a 90°, sen θ (representação algébrica de seno) variará de 0 a 1. No segundo quadrante da circunferência, sen θ também será positivo, mas decrescerá de um máximo de 1 até 0, à medida que θ for se aproximando de 180°. Todos os ângulos θ maiores que 180° até (mas não inclusive) 360° terão senos negativos.

Outra função trigonométrica muito utilizada é a tangente, definida como a relação entre o seno e o co-seno, o que equivale à razão entre o cateto oposto e o adjacente ao ângulo.

CIRCUITOS E COMPONENTES

Uma equação muito simples relaciona as três grandezas principais que entram no jogo algébrico dos circuitos elétricos e constituem a base para conhecer as funções dos componentes dos micros.

Georg Ohm, físico alemão que viveu no século XIX, foi o primeiro a deduzir uma equação matemática que relacionasse voltagem, corrente e resistência num circuito elétrico. Essa relação, chamada lei de Ohm, estabeleceu que a voltagem (tensão elétrica ou diferença de potencial) entre dois pontos de um circuito elétrico é igual ao produto da corrente pela resistência. Dessa forma, conhecendo duas das grandezas citadas, torna-se possível determinar a terceira: obtemos o valor da resistência dividindo a voltagem pela corrente; e calculamos o valor da corrente dividindo a voltagem pela resistência.

Usando os símbolos adotados por convenção, V para voltagem, I para corrente e R para resistência, essas relações podem ser representadas algebricamente de forma bastante simples:

$V = R \times I$
$R = V/I$
$I = V/R$

Uma extensão da lei de Ohm é a lei da potência:

$W = V \times I$

que relaciona W, a potência consumida num circuito ou dispositivo, com a voltagem entre seus terminais e a corrente que flui através dele. A potência é a energia despendida na unidade de tempo. Se a voltagem na fórmula anterior for expressa em volts e a corrente em ampères, a potência será calculada em watts.

O conhecimento dessas equações é indispensável para os que trabalham com circuitos elétricos. Mesmo em nível doméstico mostram-se de grande utilidade, pois permitem, por exemplo, determinar o tipo de fusível necessário a qualquer aparelho.

Como ilustração, considere um aquecedor elétrico que funcione em 220 V. Dividindo 3.000 (a potência, em watts, consumida pelo aparelho e indicada pelo fabricante) pela voltagem, a intensidade de corrente que atravessará o circuito será de 13,6 A. Desse modo, qualquer fusível de valor menor que 13,6 A se queimará imediatamente. Se o valor máximo suportável pela tomada for de 14 A, não se deve ligar a essa tomada nenhum aparelho que consuma mais do que 88 W — isto é, $(14 - 13{,}6) \times 220$ —, sob risco de comprometer todo o circuito.

Resistência e corrente

Num circuito elétrico, todos os componentes opõem resistência ao fluxo de eletricidade, da mesma forma que, num sistema de aquecimento central, os canos e reservatórios resistem à passagem da água: quanto maior a resistência apresentada, menor o fluxo. De um balde com um pequeno furo pouca água escapa, porque as bordas do furo resistem ao fluxo; se o furo for maior, a resistência será menor e, em conseqüência, sairá mais água. Nos circuitos elétricos, o processo é semelhante: os de alta resistência conduzem corrente de menor intensidade que os de baixa resistência. A resistência de um componente elétrico depende do material que o constitui. Em geral, os fios são feitos de cobre, que oferece pouca resistência.

Cuidados básicos

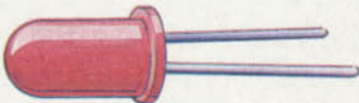
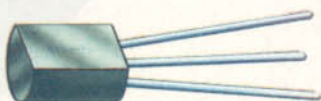
Ao fazer qualquer reparo, lembre-se de desconectar o micro da tomada. Tome o cuidado também de anotar a posição inicial dos fios e dos componentes, a fim de evitar engano na hora da remontagem. Você pode marcar na própria placa utilizando caneta dermatográfica. No caso de um cabo multivias, por exemplo, marque a posição da faixa vermelha, indicadora do fio n.º 1. Para limpar os componentes e conectores tipo pente, use álcool isopropílico.

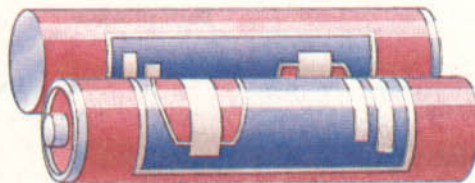
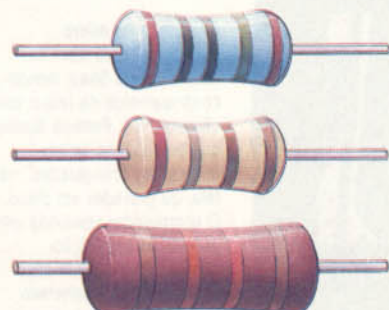
CUIDADO: seu computador perderá a garantia quando aberto por pessoas não credenciadas pelo fabricante.

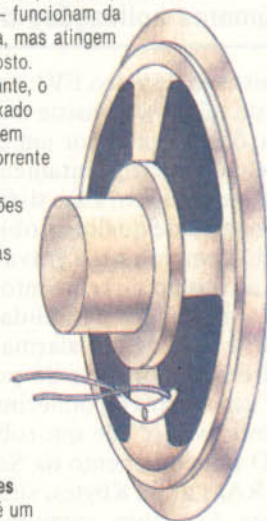
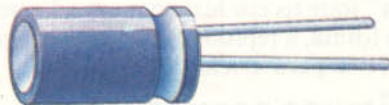
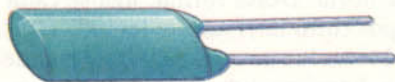
Componentes

Os diagramas de circuitos eletrônicos parecem complicados. Mas, na verdade, constituem a forma mais simples de descrever os circuitos usando um conjunto padrão de símbolos e técnicas. Neste quadro, encontram-se todos os componentes de um intercomunicador e, ao centro, seu diagrama de circuito. Cada componente, nesse diagrama, é representado por um símbolo especial e identificado por um código: R1, TR2, C3. As linhas representam fios, ou trilhas numa placa de circuito impresso.

1. Chaves
Há dois tipos principais de chave: a permanente, cujo estado se mantém após ser pressionada, e a de pressão, em que o contato só permanece enquanto se mantiver a pressão.

2. Diodos emissores de luz (LED)
Semicondutores de tipo mais simples, os diodos são o equivalente eletrônico das válvulas de retenção — só permitem a passagem da corrente num sentido. Alguns diodos, revestidos com resinas translúcidas em vez de metal, são usados como indicadores, pois emitem pequena quantidade de luz.

3. Transistores
Dependendo do tipo e da aplicação, o transistor pode agir como chave ou como amplificador. Sua invenção, em 1947, abriu o caminho para outros desenvolvimentos na microeletrônica.

4. Resistores
Introduzidos nos circuitos, servem para controlar o fluxo de eletricidade. Há resistores de diversos tamanhos e seus valores são expressos em faixas coloridas.

5. Resistores variáveis
Nem todos os resistores têm resistência constante. Alguns resistores variáveis, chamados de potenciômetros, usam uma tira de carbono como material condutor. O percurso da corrente pelo carbono determina a resistência do componente.

6. Pilhas
A maior parte dos circuitos pequenos, como o deste intercomunicador, pode funcionar com pilhas comuns, porque o consumo em watts do sistema é baixo e elas fornecem uma corrente contínua estável.

7. Alto-falantes
Alto-falantes são parentes próximos dos microfones — funcionam da mesma forma, mas atingem resultado oposto. Num alto-falante, o diagrama afixado ao ímã vibra em resposta à corrente aplicada. Essas vibrações produzem ondas sonoras no ar.

8. Capacitores
O capacitor é um dispositivo capaz de armazenar carga elétrica. Fica carregado quando se ligam seus terminais a uma fonte de energia e assim permanece mesmo depois de desligada a fonte. Para descarregá-lo é necessário conectar seus terminais entre si ou a outros condutores.

9. Microfones
Os microfones funcionam de maneira oposta à dos alto-falantes. Ondas sonoras fazem o diagrama vibrar e, em conseqüência, o ímã produz diferentes voltagens no circuito.

CÂMARA ESCURA

Do tamanho de um maço de cigarros e de fácil montagem, o sistema de vídeo EV1 reproduz na tela do computador as imagens captadas por uma câmara, proporcionando inúmeras aplicações de utilidade.

O sistema de vídeo EV1, também chamado Snap (instantâneo), consiste numa pequena câmara eletrônica ligada por um cabo à entrada em paralelo do micro, juntamente com o software em cassete ou disco. Esse sistema permite transferir a imagem de qualquer objeto ou cena para a tela do computador e gravá-la em disquete. Além do uso como divertimento, a câmara Snap serve para aplicações de utilidade prática: pode formar a base de um alarma "inteligente" contra ladrões — por meio do programa Secure — ou ser usada para reconhecimento de imagens e até como "visão" de um robô.

O funcionamento da Snap se deve a um chip de RAM de 32 Kbytes, situado no interior da câmara. Esse chip, com uma janela transparente, compõe-se de células que têm a propriedade de perder aos poucos a carga nelas armazenada, uma vez exposta à luz. A velocidade com que se descarregam é proporcional à intensidade da luz incidente. Na câmara Snap todas as células são, de início, completamente carregadas, gravando-se um valor específico em todas as posições de memória. Dessa forma, liga-se cada uma delas. Após curto intervalo de tempo, as células que receberem luz passam a descarregar-se. O programa imprime um ponto luminoso na tela na posição correspondente a cada célula "desligada" (que recebe luz) na matriz de memória. Dessa forma, a reprodução da imagem no chip transfere-se para a tela do micro.

Figuras complexas

O intervalo de tempo que o computador gasta ligando todas as células e depois lendo-as determina a "exposição" da imagem. A câmara Snap pode produzir várias imagens por segundo, quando a iluminação é intensa o suficiente; sob iluminação ambiente normal, obtém-se uma imagem fotográfica em menos de 1 segundo. A resolução da figura fica limitada pela matriz de 256 x 128 células de memória no chip da câmara.

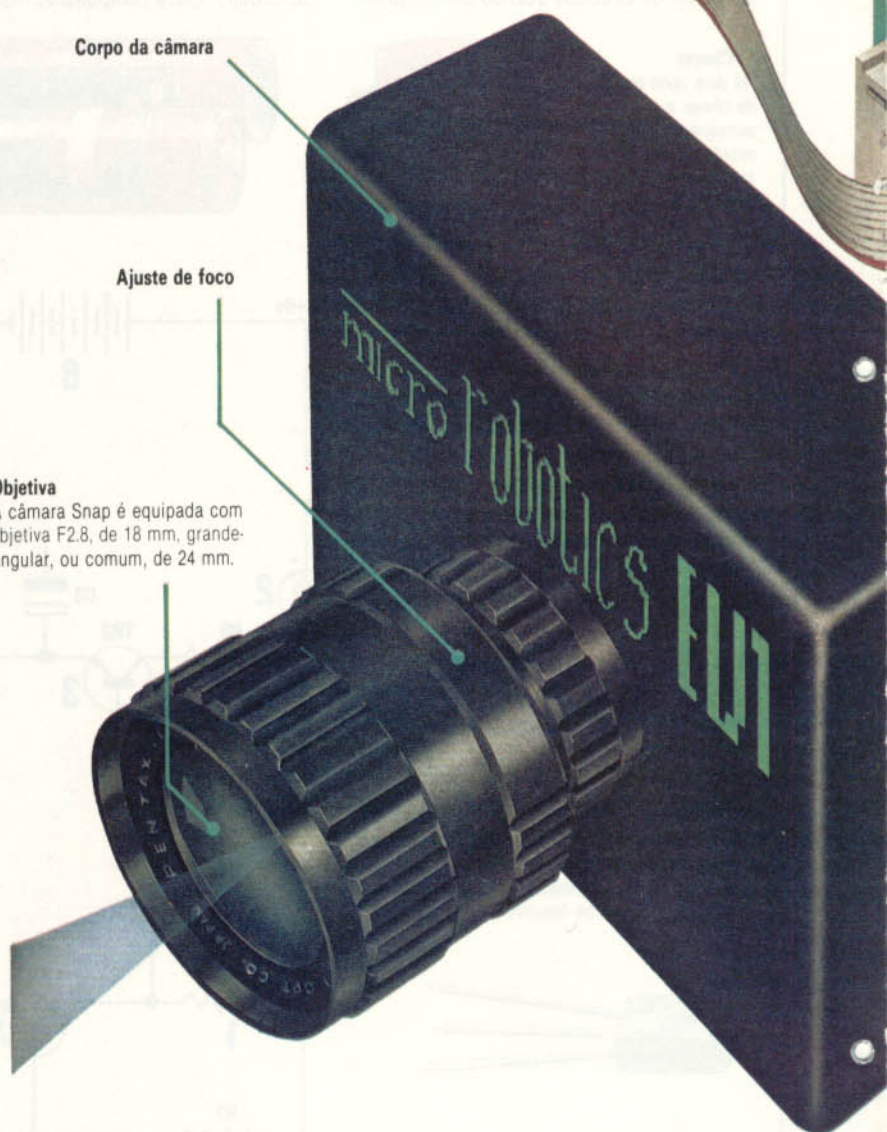
A Snap usa a objetiva de uma pequena câmara reflex Pentax 110, acomodada sobre uma armação tipo baioneta. Com ela podem-se produzir figuras de dois tons a partir de uma só

Cabo para o computador
Com 2 m de comprimento, conecta a câmara à saída do micro.

Corpo da câmara

Ajuste de foco

Objetiva
A câmara Snap é equipada com objetiva F2.8, de 18 mm, grande-angular, ou comum, de 24 mm.

micro robotics EV1

Animação no micro
O EV1, software que acompanha a Snap, mostra continuamente na tela o que a câmara "vê". Permite também que as imagens sejam impressas, "congeladas" na tela, ou gravadas em disco. O computador seleciona um período de exposição conveniente, embora esse tempo possa ser alterado pressionando-se as teclas do cursor.

CÂMARA SNAP/EV1
Dimensões: 70 x 50 x 25 mm.
Objetiva: Lentes de 24 ou 18 mm.
Interface: Paralela.
Documentação: Manual de cinqüenta páginas, explicando o funcionamento da câmara e a utilização dos aplicativos.

Resolução vertical
As figuras geradas pela Snap são formadas por linhas verticais, com lacunas que indicam variações na luminosidade. A resolução das imagens é apenas aceitável.

imagem, ou de vários tons, tirando-se diversas fotos em exposições diferentes — caso em que se obtêm figuras mais realistas.

O primeiro método é usado por um dos programas fornecidos com o aplicativo, que inclui também uma rotina de dump de tela apropriada para impressoras Epson ou compatíveis.

O programa Movie armazena uma pequena seqüência de imagens em dois tons que são mostradas rapidamente, dando o efeito de animação. Talvez a opção mais atraente seja o programa chamado Arty, usado para construir figuras complexas a partir de diferentes imagens. Com um joystick, as imagens captadas pela câmara podem ser posicionadas em qualquer lugar da tela, tendo seu tamanho ampliado ou reduzido. As cores da imagem e do fundo são mudadas por meio das teclas de função.

Claro-escuro
As figuras aparecem geralmente em preto e branco. O programa Grey permite compor as imagens em tonalidades cinza, usando oito variações de brilho. O aumento do contraste por meio do sombreado oferece mais realismo à imagem.

A PRODUÇÃO DE SOFTWARES

Muitos amadores sonham desenvolver programas campeões de vendas. Mas a concorrência é forte demais: empresas fabricantes de softwares contam com enormes recursos técnicos para criar seus best sellers.

Um equipamento caro não implica necessariamente programas de sucesso; alguns amadores conseguiram muito dinheiro com softwares produzidos em casa, num micro de poucos recursos.

Ainda assim, os gênios da programação doméstica estão se tornando uma espécie ameaçada de extinção, sobretudo com o desenvolvimento das grandes empresas de software (software houses) na década de 80. Os poderosos computadores e sofisticados recursos de programação dessas empresas representam uma vantagem real sobre o proprietário de micro doméstico e possibilitam a seus programadores mostrar-se mais produtivos.

Uma característica importante do software para microcomputadores é a velocidade de execução. Isso significa que os programas precisam ser escritos (pelo menos parcialmente) em código de máquina (ASSEMBLY). Mas é difícil trabalhar com código de máquina — e os programadores de ASSEMBLY precisam de outros softwares para ajudá-los a escrever os programas. O mínimo que se exige é um programa assembler (assemblador ou montador) que traduza o código-fonte para o código-objeto que a máquina entende. Essa será uma tarefa difícil, se o programa em questão for grande.

Os assembladores para micros não se revelaram muito eficientes. Mesmo o mais simples desses pacotes usa quantidade considerável de memória, limitando o tamanho dos programas que podem ser escritos com ele.

Por outro lado, o trabalho prolongado em certos microcomputadores resulta em desconforto e cansaço: teclados, telas rudimentares e, em alguns casos, falta de unidades de disco podem transformar o uso desse equipamento numa tarefa árdua.

Por essas razões, a maioria das empresas profissionais não desenvolve programas no próprio micro para o qual o programa foi feito (chamado de "máquina-alvo"), mas sim em computadores profissionais com software especial (os sistemas de desenvolvimento). Os programadores que usam tais máquinas costumam escrever em versões de PASCAL e C, conhecidas como

Processamento do pensamento

A idéia de um programa é desenvolvida num sistema especial para esse fim — um IBM PC, por exemplo —, que engloba as bibliotecas de rotinas mais usadas, recursos para depuração e programas de simulação. Estes imitam o sistema operacional e a tela da máquina-alvo.
A versão desenvolvida passa para o código da máquina-alvo por compilação cruzada e é então otimizada em elegância e velocidade apropriadas.

"compiladores cruzados" ou "assembladores cruzados".

Conseguem, assim, desenvolver num micro que use um processador 8086, por exemplo, programas que serão executados em máquinas com processadores Z80. Numa fase posterior, especialistas em código de máquina revêem e melhoram os programas assim desenvolvidos.

Softwares inteligentes

Rotinas de depuração de erros são acrescentadas sem a preocupação de o programa não caber na memória.

Algumas empresas européias podem exemplificar a técnica de desenvolvimento de programas. Uma delas é a Intelligent Software (IS), fundada em 1981, resultante de uma sociedade entre David Levy, especialista em xadrez, e a ANT Microwave, de Robert Madge. A IS especializou-se em jogos de estratégia, dos quais a maioria é escrita sob contrato para os micros mais populares. Desenvolveu também o software para máquinas que jogam xadrez.

Embora não haja telas de ação em jogos como o xadrez e o bridge, muita computação está envolvida em seu funcionamento. Assim como acontece com os jogos de fliperama, os de estratégia precisam da velocidade do software escrito.

Além de usar as próprias máquinas-alvo para desenvolver os programas, a IS vale-se de micros IBM PC e Apple com interfaces especialmente desenvolvidas para transferir programas de uma máquina para outra. A empresa está constantemente envolvida em projetos de conversão — adaptando, por exemplo, um jogo de xadrez de um tipo de computador para outro —, e assim é possível programar de forma modular. Por exemplo, um nível de segmentação útil quando se tem de converter programas de jogos de um processador para outro é a divisão do programa em rotinas de jogo e rotinas de entrada/saída. Estas, na máquina que está recebendo o programa, têm endereços de memória diferentes e talvez também sejam diferentes em termos de endereçamento de periféricos. Rotinas de jogos, por serem independentes do hardware (exceto o processador), têm conversão direta.

A Intelligent Software evita cercear a criatividade dos programadores; assim, há poucas "regras da casa" direcionando a programação. Um ponto importante no qual ela insiste, contudo, é que o programa-fonte deve incluir numerosos comentários, de modo que seja sempre claro o que faz cada rotina.

Quando os programadores trabalham em suas casas para uma empresa de software, cada um desenvolvendo projeto próprio, há muito pouco compartilhamento de recursos. Em decorrência, a individualidade preserva-se às custas da multiplicação de esforços, pois rotinas semelhantes acabam sendo escritas e reescritas por vários programadores.

Hardware pesado

Outra empresa de software, a Psion, utiliza computadores ainda maiores que o IBM PC. Entre as empresas de software inglesas que desenvolvem jogos para microcomputadores, a Psion é a única a concentrar suas pesquisas de desenvolvimento em minicomputadores.

Ela começou desenvolvendo softwares para o ZX81, um pequeno micro da Sinclair — e usava o próprio ZX81 para fazê-los. Para criar a fita Horizons, que acompanha o Spectrum, a Psion comprou um TRS-80 com discos, que usa o mesmo processador Z80, e criou uma interface especial entre as duas máquinas.

Mas, em agosto de 1982, a empresa decidiu que não poderia continuar criando sistemas de desenvolvimento completamente diferentes cada

Módulos universais
O desenvolvimento de jogos de estratégia (o xadrez, por exemplo) é feito na máquina-alvo ou em máquinas mais poderosas, como o IBM PC. Programas divididos em módulos universais podem ser facilmente adaptados de um computador para outro

Uso da máquina-alvo
A partir da definição da estratégia e do roteiro do jogo na máquina-alvo, o programa é desenvolvido em ASSEMBLY.

vez que um novo microcomputador era lançado no mercado. Então, aplicou recursos na compra de hardware pesado com bastante capacidade de processamento. Em princípio, esses hardwares deveriam ser flexíveis o bastante para enfrentar qualquer computador que aparecesse no futuro. As máquinas escolhidas foram dois minicomputadores Vax 750, que usam o sistema operacional DEC (Digital Equipment Corp) em VMS (Virtual Machine System).

Os Vax 750 trouxeram duas vantagens para a Psion: a qualidade de software proporcionada pelo DEC, com a oportunidade que ele dá de criar recursos auxiliares de software especialmente projetados, e a "força bruta" da combinação do sistema operacional com o hardware. Há muito espaço para uma coleção de recursos de software, como compiladores, bibliotecas de sub-rotinas comuns e programas de depuração, todos compartilhados até entre vinte programadores, que podem estar acessando uma única máquina ao mesmo tempo. As duas máquinas permitem que o software seja convertido de uma para outra, quando necessário, com facilidade.

As bibliotecas de sub-rotinas comuns já faziam parte da filosofia da Psion no tempo do TRS-80, mas ficar trocando dados entre dois discos é trabalhoso. Com as máquinas Vax, equipes de programadores podem até trabalhar juntas, compartilhando bibliotecas de projetos, das quais se conseguem módulos quase instantaneamente. Essas bibliotecas são até compartilhadas por equipes ocupadas em projetos diferentes.

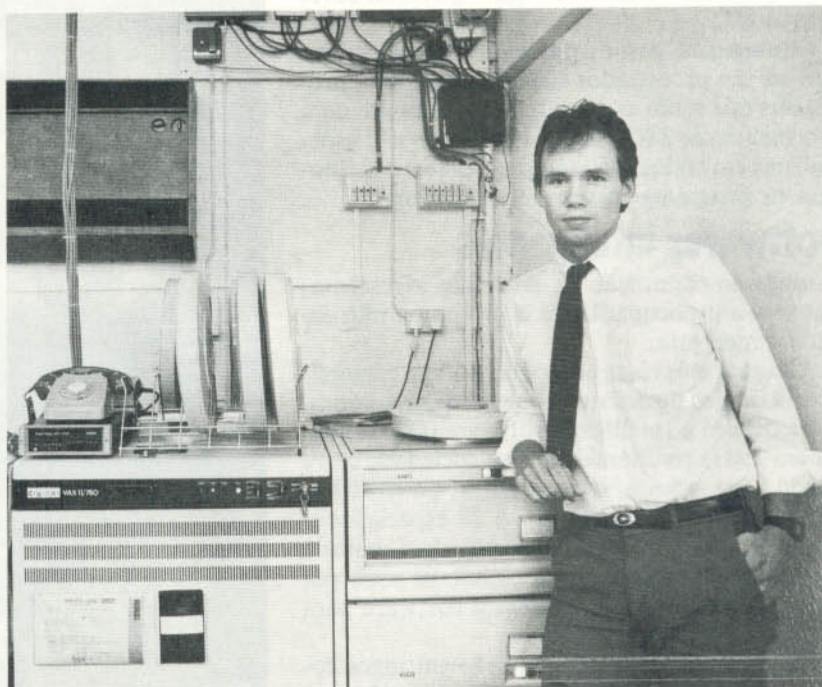
Essa é a grande vantagem de um sistema com tempo compartilhado — e, como vantagem adicional, ele também cuida do trabalho administrativo sem ter de interromper os programadores.

Mesmo se você tem condições para isso, não pense que, comprando um Vax, ficaria imediatamente em igualdade de condições com uma firma como a Psion. Uma parte muito pequena desse ambiente de trabalho bem desenvolvido foi entregue numa bandeja à Psion pelo DEC. A empresa despendeu muito esforço para conseguir que tarefas simples fossem realizadas com eficiência e confiança e para chegar ao grande número de recursos de software e utilitários elaborados à mão (escritos em C) que acumulou.

A Psion usa C, linguagem de grande portabilidade que pode produzir um código-objeto compacto e rápido para chips de 16 bits, como o 8086. Mas isso não se aplica a compiladores C de 8 bits. Assim, ao escrever para máquinas-alvo como o Spectrum, a empresa precisou desenvolver suas próprias técnicas especiais.

Sabe-se que a Psion usou C para escrever seu próprio compilador. Ele se parece um pouco com o C, é transferível para diferentes processadores e cria programas muito eficientes.

Código-fonte

Há uma regra segundo a qual a manutenção dos sistemas e o trabalho de escrever recursos auxiliares de programação (como o compilador da Psion) ocupam 30% do tempo total de programação. Mas a empresa acha que esses homens-hora a mais valeram a pena. Desenvolver seu próprio código-fonte significa ter domínio absoluto sobre ele: você pode desmontá-lo e aperfeiçoá-lo ou adaptá-lo, o que é impossível com o software adquirido no comércio.

O software especial comprado pela Psion inclui programas que são simulações exatas dos microprocessadores mais comuns, como o Z80 e o 6502. Assim, pode-se fazer com que os computadores Vax se comportem como se fossem um Commodore 64 ou um Spectrum. Apesar da grande capacidade dos computadores Vax, os simuladores fazem-nos funcionar numa fração da velocidade da máquina-alvo.

A vantagem é que eles permitem que o programador examine o conteúdo de cada registro dentro do microprocessador em qualquer etapa do programa, facilitando sobremaneira a detecção de erros. Normalmente, quando ocorre um erro num programa em código de máquina, é quase impossível dizer o que saiu errado. A Psion poupa, assim, muitas horas de depuração.

A maior parte do esforço de desenvolvimento da Psion foi canalizada para a produção do conjunto de quatro programas comerciais que vêm com o Sinclair QL. A família de chips Motorola 68000, um dos quais é responsável pelo funcionamento do QL, foi projetada com base nas linguagens de alto nível. Os programas em C são compilados com tanta eficiência, com esses chips, que se torna desnecessário escrever em ASSEMBLER. Se todos os computadores de uso doméstico seguissem o exemplo do QL, o C substituiria por completo o ASSEMBLY e a Psion e as empresas menores de software poderiam abandonar o trabalho extenuante de fazer conversões à mão.

Exceção brasileira
Softwares para micros podem resultar do trabalho de vários programadores que compartilham um equipamento maior, como o Vax 750. No Brasil, contudo, é raro o uso de "compiladores cruzados". Alguns fabricantes de micros que desenvolvem seu próprio software constituem uma exceção.

EM CONJUNTO

Além das estruturas de dados já vistas, como matrizes e arquivos, o PASCAL inclui os conjuntos, um tipo estruturado de dados que será examinado a seguir, juntamente com os níveis de prioridade dos operadores.

Um conjunto nada mais é do que uma coletânea de objetos processados em bloco, como uma entidade singular, em vez de cada elemento ser acessado individualmente. Um exemplo seria o conjunto das pessoas que programam em PASCAL ou as letras do alfabeto que correspondem às vogais. Nem sempre os elementos de um conjunto estão ordenados e, nesse caso, a única questão relevante é saber se determinado objeto pertence ou não ao conjunto.

Para aumentar sua eficiência, o PASCAL impõe algumas restrições aos conjuntos. Seus membros só podem ser tipos escalares simples — e não matrizes ou outros dados estruturados — e há um limite máximo (definido pela implementação) para esse tipo "base". A sintaxe é simples:

```
TYPE
  NUMEROS = SET OF 0 .. 127;
  ALFAS = SET OF 'A' .. 'Z';
  CORES = SET OF (VERMELHO, VERDE,
    AZUL);
```

A faixa dos valores que o tipo base ordinal do conjunto pode assumir é definida por meio da mesma sintaxe usada para os tipos de subfaixa. As variáveis de conjunto são declaradas na parte da declaração VAR, do mesmo modo que as outras variáveis do PASCAL; e podem, nas instruções, vir por extenso entre colchetes (isto é, como "literais"):

```
VAR
  CODIGOS : NUMEROS
  LETRAS : ALFAS
BEGIN
  CODIGOS := [0 .. 2, 4, 8, 16, 32, 64];
  LETRAS := ['A' .. 'Z']; {ETC.}
```

Esse segmento inicializaria os códigos do conjunto, fazendo-os conter somente os números de 0 a 2 (inclusive), de 4 a 8, e assim sucessivamente. Como não há uma ordem inerente ao próprio conjunto (ao contrário de seu tipo base), poderíamos da mesma forma representar esse conjunto como [64, 32, 16, 8, 4, 0 .. 2], mas observe que a subfaixa 2 .. 0 (ilegítima na presente definição de subfaixa) iria meramente indicar uma faixa vazia. Qualquer conjunto pode ser inicializado como um conjunto vazio de seu tipo por meio da instrução QUALQUERSET := [], que origina a única exceção à regra geral do PASCAL de que se conhece por inspeção o tipo de qualquer literal. Se pelo menos um membro do conjunto não aparecer por extenso, não poderemos — nem o compilador — determinar seu tipo. Felizmente, o conjunto vazio pode ocorrer somente em atribuições (como no exemplo) ou em expressões nas quais os outros identificadores já foram declarados. Isso significa apenas que um conjunto vazio sempre é um subconjunto de qualquer conjunto.

Operadores de conjunto

Um dos operadores mais úteis de que o PASCAL dispõe para estruturas de conjuntos é, como DIV e MOD, uma palavra reservada: IN. Trata-se de um operador que relaciona dois operandos e verifica a presença de um elemento num conjunto específico. O lado da esquerda deve ser uma expressão que resulta em um dos elementos possíveis do conjunto — em outros termos, um valor do tipo base do conjunto — e o lado da direita pode ser uma variável de conjunto ou uma literal de conjunto. Toda a expressão dará um resultado booleano: verdadeiro, se o valor for membro do conjunto, e falso, em caso contrário.

Assim: N IN CODIGOS e X IN LETRAS são expressões booleanas legítimas. Para verificar se um caractere é um dígito, o teste

```
IF (C > = '0') AND (C < = '9') THEN...
```

é muito simplificado se testamos o valor de C, examinando se ele é membro do conjunto de caracteres em que estamos interessados:

```
IF C IN ('0' .. '9') THEN...
```

Ou, talvez, se estivermos desenvolvendo um programa para jogar cartas:

```
TYPE
  ROL = (DOIS, TRES, QUATRO, CINCO, SEIS,
    SETE, OITO, NOVE, DEZ, VALETE, DAMA,
      REI, AS);
  CARTAS = SET OF ROL
VAR
  CARTA : ROL
  FIGURAS : CARTAS
BEGIN
  FIGURAS : VALETE..AS);
  IF CARTA IN FIGURAS THEN {ETC.}
```

Compare esse exemplo com:

```
IF (CARTA = VALETE) OR (CARTA = DAMA)
  OR (CARTA = REI) OR...
```

No PASCAL também se definem operações envolvendo conjuntos inteiros: interseção, união e diferença de conjuntos. Se B for o conjunto de todos os programadores em BASIC e P representar os programadores em PASCAL, a interseção dos dois conjuntos será o conjunto de programadores que usam tanto BASIC como PASCAL. A união de P e B é o conjunto de pessoas que programam ou em PASCAL ou em BASIC.

Tacada perfeita

O jogo de sinuca pode ser disputado com quinze bolas vermelhas (cada uma valendo 1 ponto), uma bola "jogadeira" branca e seis coloridas, que equivalem aos seguintes pontos: amarela, 2; verde, 3; marrom, 4; azul, 5; rosa, 6; e preta, 7.

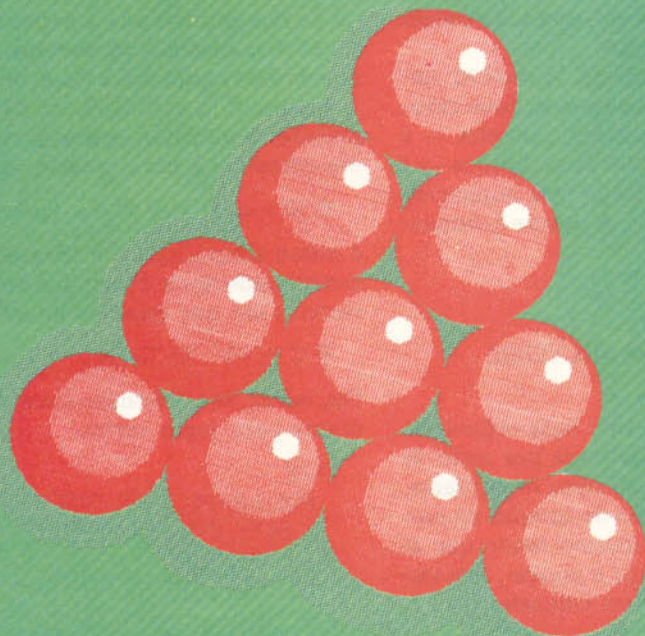
Após cada bola vermelha ser encaçapada, o jogador pode testar sua habilidade com uma das outras cores. Estas, se encaçapadas, são a seguir recolocadas na mesa. Depois da última combinação de vermelho e outra cor, todas as cores devem ser encaçapadas em ordem crescente de valor.

Na sinuca, uma série de tacadas só termina: **a)** quando o jogador deixa de encaçapar uma bola; **b)** quando comete alguma falta; e **c)** quando não resta nenhuma bola (encerrando-se o jogo). Escreva um programa para calcular a maior série possível de tacadas, mas — e aqui está o truque — de tal modo que 15 seja o único número existente no programa.

A solução será apresentada na próxima matéria desta seção.

Prioridade de operadores

Esta tabela mostra a ordem de prioridade dos operadores do PASCAL. Se necessário, o uso de parênteses obriga à avaliação separada da expressão por eles delimitada, cancelando a ordem normal de prioridade.

Prioridade	Operadores	Tipo
Máxima	NOT + -	(monádica)
	AND * /DIV MOD	(multiplicação)
	OR + -	(soma)
Mínima	< <= <> >= > IN	(relacional)

A diferença de conjuntos é, como o nome sugere, o resultado de se retirar ou subtrair de um conjunto todos os membros de outro conjunto. Assim, na notação do PASCAL, P-B representa todos os que programam em PASCAL mas não em BASIC. Analogamente, a notação usada para a união será P+B, e para a interseção, P * B. Esses operadores são indicados pelos mesmos símbolos empregados para os operadores aritméticos, pois, nas operações com conjuntos, os operadores representam o teste dos bits envolvidos.

Imagine um conjunto de oito elementos. A presença de determinado elemento num conjunto poderia ser indicada colocando-se o bit apropriado em um grupo de 8 bits (1 byte); enquanto sua ausência no conjunto seria assinalada no byte por meio de um zero. Esses testes de inclusão exigem apenas uma verificação de bits: o conjunto união resulta de uma operação OR; e a interseção, de um simples AND. Essas operações são efetuadas em nível de código de máquina, de modo que o compilador do PASCAL possa fornecer estruturas de dados de conjuntos e suas operações correspondentes. O dispêndio geral de memória é mínimo e, quando o tamanho dos conjuntos pode ser mantido no limite do tamanho das palavras do computador (em máquinas de 32 bits ou mais, por exemplo), as operações com conjuntos são as desempenhadas com mais eficiência pelo PASCAL.

Ordem de prioridade

Podemos agora resumir todos os operadores usados em PASCAL. Aqueles que não foram citados são todos conhecidos de outras linguagens, mas no PASCAL tudo é simplificado, pois existem apenas quatro níveis de prioridade de operadores. Evidentemente, os operadores "monádicos" têm prioridade sobre os demais. São os símbolos + e -, que indicam o sinal dos números, e o operador de negação booleana NOT. O segundo nível de prioridade inclui todos os operadores de "multiplicação" (inclusive os símbolos de divisão) seguidos pelos de adição e subtração e, por fim, no nível mais baixo de prioridade, todos os operadores relacionais, inclusive IN.

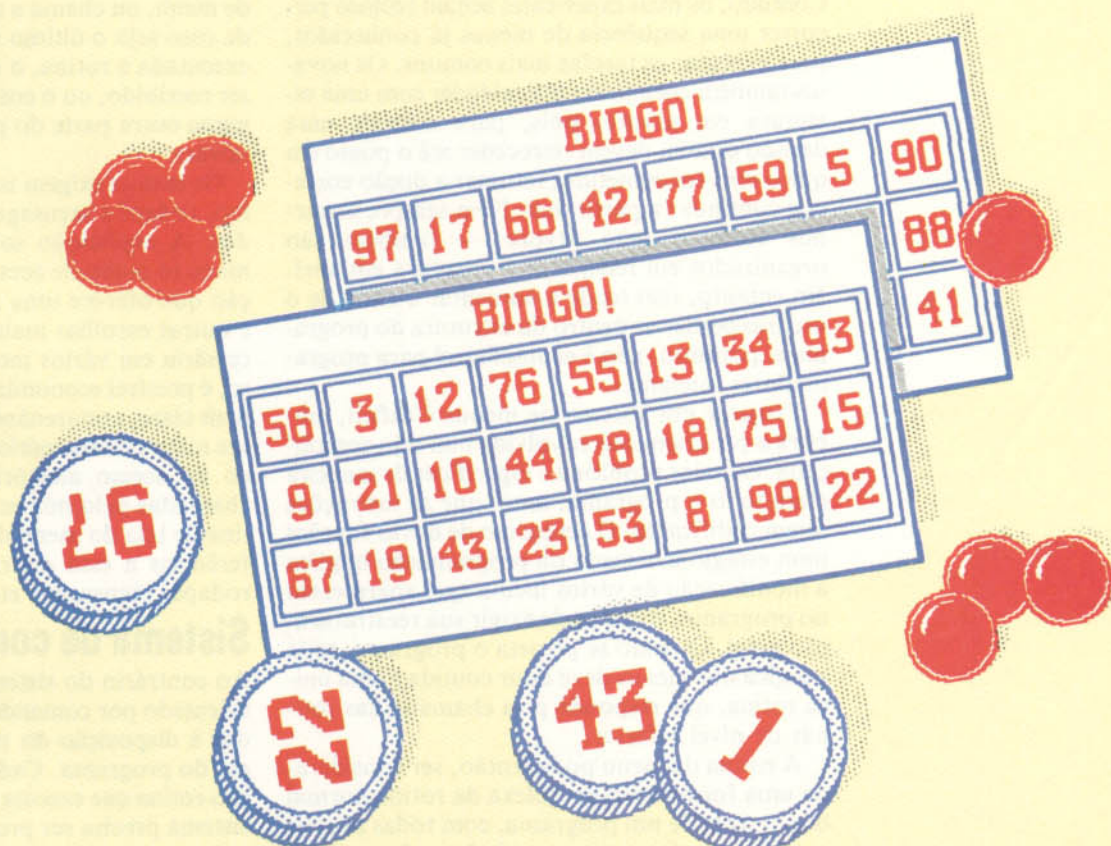
Observe que os dois outros operadores booleanos (AND e OR) são operadores de multiplicação e de adição, respectivamente. Isso reflete a álgebra booleana utilizada e implica a colocação de muitos testes relacionais entre sinais delimitadores para cancelar essa prioridade. Caso contrário, IF N>0 AND N<10 THEN..., por exemplo, resultará em erro quando o programa for compilado, pois a expressão 0 AND N (avaliada em primeiro lugar) tenta combinar dois operandos integer com um operador booleano.

Quaisquer operadores com o mesmo nível de prioridade são avaliados, como de costume, da esquerda para a direita. O símbolo de atribuição (:=) possui uma prioridade mais baixa que qualquer outro dos operadores acima, pois a expressão do lado direito deve ser completamente avaliada antes de se fazer a atribuição.

Bingo!

Um conjunto é o melhor modo de se representar uma cartela de bingo. Embora os elementos (integers de 1 a 90) sejam ordenados, a única questão relevante é saber se o número chamado está ou não na cartela ou, em PASCAL, se "NUMERO IN CARTELA" é verdadeiro ou falso.

O programa simula um jogo de bingo lendo os números da cartela por meio do teclado e, depois, à medida que são chamados, verificando se correspondem ao conjunto da cartela. A expressão CARTELA – CHAMADOS será um conjunto vazio quando todos os números de CARTELA também estiverem no conjunto CHAMADOS. Note que não se pode incluir um membro diretamente no conjunto; só pela criação de um conjunto com um único elemento (colocando o número entre colchetes) podemos efetuar a união de dois conjuntos. Assim como está, o programa admitirá entradas duplicadas de cartelas e números fora da faixa de 1 a 90, causando um erro de execução. Como exercício, imagine um método para acrescentar estruturas que evitem essas duas irregularidades.

```
PROGRAM    BINGO    (INPUT, OUTPUT);

CONST
       COLUNAS = 40; {PARA CONTROLAR VIDEO}
       METADE = 25; {IDEM}

TYPE
       BINGO = SET OF 1 .. 90;

VAR
       CONTADOR          : 1 .. 15;
       POSSIVEIS,
       CHAMADOS,
       VAZIA,
       CARTELA           : BINGO;
       NUMERO            : INTEGER;
       FEITO             : BOOLEAN;

BEGIN

  VAZIA := [];
  POSSIVEIS := [1 .. 90]
  WRITELN ('*** BINGO ***' : METADE);
  WRITELN;
  WRITELN ('ENTRE COM OS 15 NUMEROS DA
    CARTELA');
  WRITELN ('(CADA UM SEGUIDO DE RETURN) : ');
  CARTELA := VAZIA;

  FOR CONTADOR := 1 TO 15 DO
    BEGIN
      WRITE (CONTADOR : 10, ' : ?');
      READLN (NUMERO);
      CARTELA := CARTELA + [NUMERO]
    END;

  WRITELN;
  WRITELN ('OLHO VIVO!' : METADE);
  WRITELN;
  WRITELN ('AGORA CHAME OS NUMEROS : ');
  CHAMADOS := VAZIA;

  REPEAT
    WRITE ('?' : METADE);
    READLN (NUMERO);
    CHAMADOS := CHAMADOS + [NUMERO];
    FEITO := CARTELA - CHAMADOS = VAZIA

  UNTIL FEITO;

  WRITELN;
  WRITELN ('PARABENS!' : METADE);
  WRITELN ('SEUS NUMEROS ERAM:');
  WRITELN;

  FOR NUMERO := 1 TO 90 DO
    IF NUMERO IN CARTELA THEN
      WRITE (NUMERO : COLUNAS DIV 8)

END.
```

FAÇA SUA ESCOLHA

Qualquer programa complexo exige que o usuário escolha, em determinados pontos, entre vários caminhos alternativos. Os sistemas orientados por menus ou por comandos são recursos que facilitam essas decisões.

Um menu pode ser tanto uma simples lista de itens numerados como uma tela cheia de complexos símbolos visuais, mas o princípio que rege seu uso é sempre o mesmo. Emprega-se o menu quando a lógica do programa chega a um ponto em que são possíveis vários caminhos alternativos e o usuário escolhe, a partir de uma lista de opções mostrada na tela, qual a rota a ser tomada. Os programas baseados em menus possuem, em geral, uma estrutura em forma de árvore: o usuário entra na árvore pela "raiz" e é guiado, pelas opções do menu, até uma das "folhas", onde a informação ou função específica se encontra.

A vantagem disso é não exigir do usuário nenhum conhecimento do programa, porque a rota é bem "sinalizada" em toda a sua extensão. Contudo, os mais experientes acham tedioso percorrer uma seqüência de menus já conhecidos, para executar as tarefas mais comuns. Os novatos também encontram dificuldades com uma estrutura em árvore, pois, para corrigir uma decisão errada, devem retroceder até o ponto em que o erro foi cometido, retomar a opção correta e continuar a partir dali. Nem sempre os menus formam uma árvore — também são organizados em rede, numa estrutura em anel. No entanto, isso tende a aumentar o risco de o usuário perder-se dentro da estrutura do programa e, portanto, não é aconselhável para programadores iniciantes.

Projetar um sistema de menus é difícil, embora a programação envolvida não seja complicada. O maior problema é especificar de maneira clara todo o programa, antes que as instruções sejam codificadas. O acréscimo de novas funções num estágio avançado da programação implica a modificação de vários menus que apareceram no programa, o que pode exigir sua reestruturação geral. Quando se projeta o programa, toda a lógica dos menus deve estar contida numa única rotina, que responde pela chamada das rotinas de nível inferior.

A rotina de menu pode, então, ser considerada uma forma mais complexa da rotina normal de controle de um programa, com todas as suas ramificações internas controladas pelo usuário. Isso assegura um bom nível de organização para o projeto e serve para separar a lógica de controle e as partes funcionais do programa, permitindo que cada uma seja desenvolvida e depurada em separado.

O fluxo de controle vai agora seguir um caminho predeterminado. Para cada menu ao longo da rota, a rotina da lógica do menu transfere um conjunto de mensagens ao usuário para outra rotina que as encaixa na "moldura" do menu. Em geral, este contém uma mensagem de cabeçalho, que mostra um título e quaisquer outras informações necessárias sobre o menu; um "rodapé", que explica como fazer a escolha (com espaço suficiente para a resposta do usuário); e as opções do menu propriamente ditas. Uma tela de menu eficiente tem no máximo oito opções dispostas em coluna, com os códigos de resposta (número, letras, código mnemônico etc.) à esquerda de cada item.

A rotina de menu chama uma rotina de entrada, passando-lhe as condições para entradas válidas, e aceita a resposta do usuário. Em seguida, a rotina de menu interpreta essa resposta (que em geral é dada pressionando-se uma única tecla) e passa o controle para o próximo nível de menu, ou chama a rotina aplicativa apropriada caso seja o último menu da cadeia. Uma vez executada a rotina, o menu que a chamou pode ser reexibido, ou o controle ,pode passar para alguma outra parte do programa (o menu inicial, talvez).

Os menus exigem muito texto para o cabeçalho, rodapé e mensagens, e muitos serão repetidos. A explicação sobre o funcionamento do menu (o modo de acesso às telas de ajuda, a opção que oferece uma saída para o menu inicial e outras escolhas mais freqüentes) pode ser necessária em vários menus diferentes. Nesse caso, é possível economizar espaço e tornar a lógica mais clara, armazenando todas as mensagens numa matriz alfanumérica (ou num arquivo em disco de acesso aleatório), da qual possam ser chamadas pelo número de índice. Projete a rotina de tela do menu de forma que ela aceite referências a essa matriz e mostre o cabeçalho, rodapé, mensagens etc. apropriados.

Sistema de comandos

Ao contrário do sistema de menus, um sistema orientado por comandos coloca todos os comandos à disposição do usuário em qualquer estágio do programa. Cada comando remete a uma sub-rotina que executa a função requisitada. Esse sistema precisa ser projetado de forma a checar todas as entradas e diferençar os dados dos co-

A la carte
O menu pode ser projetado para aplicações específicas, mas assim só será usado para um único objetivo. Se for comum a muitos programas — como mudar a combinação de cores na tela —, a rotina poderá ser incluída na biblioteca de utilitários do programador.

Às ordens
O software orientado por comandos pode ser melhorado pelo acréscimo de mensagens análogas às dos menus. Isso ocorre com alguns processadores de texto que, embora controlados por comandos, dispõem de um bom suprimento de mensagens esclarecedoras.

A estrutura do menu
Não é difícil programar o sistema convencional de menu, mais adequado para usuários novatos. Ele deve ser escrito como uma rotina de formatação da tela, cujos parâmetros são o cabeçalho, o rodapé e as opções, além de uma rotina para verificar as entradas do usuário.

mandos do programa. Assinala-se essa diferença pelo uso de determinada tecla antes de entrar com um comando. A tecla [Control] é freqüentemente usada para isso, assim como os caracteres ponto (.) e barra (/); uma vez que não são usados no início de uma frase ou palavra, servem para indicar que a palavra seguinte é um comando e não um dado ou texto.

Nos sistemas orientados por comandos, a "árvore" é muito superficial e abrangente, e uma única rotina, funcionando como programa de controle, guia o usuário até a sub-rotina requisitada. Esse "intérprete de comandos" tem quatro tarefas principais: 1) esperar por uma entrada do usuário; 2) analisar essa entrada — isto é, dividi-la em seus componentes funcionais; 3) interpretar o comando preparando o chamado à sub-rotina apropriada (Qual é o endereço da rotina? Há parâmetros que devem ser transferidos?); e, por fim, 4) chamar a rotina a ser executada. Quando o controle retorna, o intérprete retoma o início do ciclo.

O formato de um comando pode ser bastante complexo e alguns comandos são semelhantes à linguagem humana comum. Um exemplo de linguagem orientada por comandos é o sistema óperacional Unix, cujo formato típico de comando é:

Comando + lista opcional de parâmetros

Exemplo:

L

ou

L-1

Aqui, o comando L do Unix lista um diretório de arquivos, enquanto L-1 (onde -1 é um parâmetro opcional) lista um diretório de arquivos num formato "longo".

O analisador deve ser capaz de reconhecer as várias partes do comando. O Unix simplifica o procedimento, assumindo que a primeira palavra seja um comando e reconhecendo os parâmetros pelo sinal de menos que os precede. Embora inúteis para o intérprete de comandos, os parâmetros são exigidos pelas sub-rotinas por ele chamadas. As rotinas de um sistema orientado por comandos deveriam, idealmente, adotar um formato padronizado para os parâmetros de entrada. Se isso for feito, o intérprete de comandos transferirá os parâmetros na mesma forma em que foram introduzidos (como strings, por exemplo).

É muito mais fácil criar um intérprete de comandos do que desenvolver um sistema de menus. Usuários experientes tendem a preferir sistemas de comando, mais rápidos e mais flexíveis do que os programas orientados por menus. A maioria dos sistemas operacionais é orientada por comandos, o que representa uma infelicidade para os principiantes, uma vez que oferece poucas indicações e as rotinas de auxílio em tela (se existirem) exigem algum conhecimento do sistema. Além disso, o grande número de comandos e parâmetros opcionais em um típico sistema de comandos faz com que mesmo aqueles razoavelmente familiarizados com o sistema recorram ao auxílio na tela ou precisem consultar com freqüência o manual de operação.

Dois em três
Há, em geral, muitas maneiras de se solucionar o mesmo problema — estas telas mostram as diferentes opções de que um usuário dispõe para mudar as cores do fundo e das margens da tela, e também do próprio texto.

RADIOCONTATO

De origem modesta, a Motorola Incorporated é hoje uma das maiores empresas de componentes eletrônicos do mundo, com fábricas nos Estados Unidos e também na Europa.

Robert Galvin
Presidente da Motorola.

Como muitas outras empresas de sucesso, a Motorola começou com um único homem. Sua criação data de 1928, quando Paul Galvin fundou a Galvin Manufacturing Corporation, em Chicago, Estados Unidos. Ali ele se especializou na produção de rádios domésticos. Mas, nos anos 30, a empresa diversificou sua produção, fabricando rádios para a polícia e para automóveis, com a marca Motorola. Na década seguinte, a Motorola Incorporated — como já era conhecida — tornou-se uma das primeiras empresas eletrônicas a produzir semicondutores.

Paul Galvin faleceu em 1959 e seu filho Robert assumiu a presidência da empresa. Nos anos seguintes, outros fabricantes, sobretudo japoneses, começaram a competir com ela no mercado de semicondutores e da eletrônica em geral. A recessão mundial que teve lugar na metade da década de 70 provocou perdas vultosas para a Motorola, forçando-a a modificar sua estratégia. Ela contratou uma nova equipe, que incluía muitos técnicos de uma empresa rival, a Texas Instruments, e decidiu abandonar o campo da eletrônica tradicional — onde não podia mais competir —, concentrando-se na microeletrônica de alta tecnologia.

Isso implicava a venda de parte do patrimônio da Motorola (especialmente o setor de televisão em cores), o investimento de grandes somas em pesquisa e desenvolvimento e a compra de empresas em áreas novas, com o objetivo de causar impacto. O risco era considerável, mas havia pouca possibilidade de escolha.

Sede da Motorola, em Illinois, EUA

No fim da década de 70, a Motorola estava bem longe das principais empresas do mercado de semicondutores, mas, depois de altos investimentos em tecnologia, passou a ameaçar a liderança da Texas Instruments, a maior do ramo.

"As empresas concorrentes da Motorola já desapareceram porque não se adaptaram ao meio ambiente", comenta Robert Galvin.

A Motorola continuou, no entanto, a ter problemas para colocar seus produtos no mercado na época certa. Na metade da década de 70, quando a indústria do microcomputador dava os primeiros passos, o microprocessador de 8 bits 6800 da Motorola foi superado em vendas pelo chip Mostek 6502, principalmente por este ter sido adotado pela Apple para seus computadores pessoais de grande sucesso. Também o Intel 8085 e o Zilog Z80, usados em outros microcomputadores populares, impediram o êxito do 6800. Em 1976, a empresa lançou o 6809, reconhecidamente o melhor microprocessador de 8 bits. Mas a corrida pelo mercado já estava perdida e o chip só apareceu em poucos microcomputadores, como o Tandy e o Dragon.

Contudo, a empresa continuou a investir em pesquisa e em 1985 estava muito bem colocada na corrida pelo mercado de 16 bits. O microprocessador 68000 foi lançado em 1979, embora só se encontrasse distribuído no mercado em 1982. Adotaram-no os computadores Lisa e Macintosh, da Apple, e o QL, da Sinclair. Trata-se de um chip muito potente, que contém dezessete registros de 32 bits, um bus de dados de 16 bits e um bus de endereçamento de 24 bits.

A Motorola continua a desenvolver novos produtos em seus centros de pesquisa em Phoenix, Arizona (EUA), em Genebra (Suíça) e em East Kilbride (Escócia). A fábrica de East Kilbride produz chips semicondutores complementares de óxido metálico (CMOS, Complementary Metal Oxide Semiconductors) e semicondutores de óxido metálico (MOS, Metal Oxide Semiconductors) para uma grande variedade de aplicações. A empresa está organizada em cinco grupos, trabalhando com comunicações, semicondutores, sistemas de informação, eletrônica automotiva e industrial e eletrônica militar.

Apesar de haver certa preocupação com a baixa lucratividade de alguns setores, a Motorola efetuou no primeiro trimestre de 1983 vendas no valor de 1,26 bilhão de dólares.

O CÉREBRO ARTIFICIAL

Mentes mecânicas
As pesquisas de inteligência artificial visam ao projeto de sistemas capazes de realizar tarefas que sempre exigiram a inteligência humana. No entanto, seu campo ampliou-se e passou a incluir funções perceptivas, como a visão e a audição. O objetivo é a programação de máquinas que simulem aspectos do comportamento e do raciocínio humanos.

As pesquisas sobre inteligência artificial são o que de mais avançado há no campo da informática. Nesta série de artigos, apresentaremos suas técnicas básicas, que você poderá explorar com o BASIC de seu micro.

Desde a metade da década de 70, vem aumentando o prestígio das pesquisas no campo da inteligência artificial, que passaram a ser realizadas inclusive por empresas particulares. Para entendermos a importância dessas pesquisas e os rumos que tomarão no futuro, nada melhor do que considerar seu passado.

Uma história resumida dessa área da ciência da computação poderia ser dividida em quatro partes, correspondentes a décadas e temas diferentes. Essa é, sem dúvida, uma simplificação grosseira, mas deixará evidentes os pontos mais importantes do assunto. Os temas são as respostas que um pesquisador de cada década daria à pergunta: De que trata a inteligência artificial?

Década de 50: Rede de neurônios
Década de 60: Pesquisa heurística
Década de 70: Sistemas especialistas
Década de 80: Aprendizagem por máquinas

Em 1943, Warren McCulloch e Walter Pitts desenvolveram um modelo matemático do neurônio do cérebro humano e animal, que serviu de base para uma representação simbólica da atividade cerebral. A partir dessa e de idéias semelhantes, outros estudiosos — entre os quais Norbert Wiener — elaboraram o que veio a ser chamado de "cibernética", de onde se originariam, na década de 50, as pesquisas de inteligência artificial.

A formalização do neurônio por McCulloch constituiu o fundamento das primeiras pesquisas de inteligência artificial. Dada a enorme complexidade do cérebro, não surpreende que tenha fracassado a tentativa de criar sistemas inteligentes a partir desse modelo.

Como o cérebro soluciona problemas de modo inteligente, o que pareceu mais promissor foi partir para a simulação de seu funcionamento. No entanto, nem o hardware nem o software da época estavam à altura dessa tarefa.

Dos sistemas então desenvolvidos, um dos poucos que tiveram algum sucesso foi o Perceptron, de Rosenblatt. Tratava-se de um sistema capaz de reconhecer padrões visuais predeterminados. Como mostra o diagrama da página 259, o Perceptron consiste numa rede de células fotossensíveis, simulando uma retina ocular em miniatura. Além disso, possui dispositivos detectores de subpadrões visuais predefinidos — chamados "demons" (demônios) — que monitoram (verificam ininterruptamente) o estado de grupos de células na rede. Ao reagirem à presença de subpadrões característicos, eles emitem um sinal para um centro de decisões. Este multiplica cada sinal enviado por um "demônio" por um fator de ponderação, positivo ou negativo, e soma os números resultantes. Se o total ultrapassa um limiar preestabelecido, o Perceptron indica "sim" (ou seja, há equivalência das imagens); caso contrário, diz "não". Assim, o Perceptron distingue entre duas imagens, embora seja possível, aplicando os mesmos princípios, fazer com que diferencie entre mais de duas.

Quando as esperanças de que os Perceptrons pudessem resolver grande número de problemas se revelaram infundadas, os pesquisadores de inteligência artificial reconsideraram sua representação do pensamento humano, passando a vê-lo como coordenação de tarefas simples de manipulação de símbolos. Embora implicando mudança radical na orientação das pesquisas, isso significou que os projetos começaram a ser desenvolvidos sobre bases mais firmes. Afinal, os computadores já desempenhavam tarefas como a pesquisa e comparação de símbolos, consideradas o fundamento da resolução inteligente de problemas. A grande dificuldade era juntar essas atividades simples.

Os pesquisadores mais influentes da década de 60 foram Alan Newell e Herbert Simon, da Universidade Carnegie-Mellon, que trabalhavam com sistemas para a demonstração de teoremas e com jogos de xadrez por computador, entre outras investigações. A contribuição mais importante dos dois foi o programa Solucionador Geral de Problemas (em inglês, General Problem Solver — GPS). Era geral porque o usuário definia um "ambiente de tarefas", de acordo com os objetos de determinada área de estudos e com os operadores aplicáveis a esses objetos. No entanto, a generalidade limitava-se a problemas envolvendo um conjunto relativamente pequeno de estados e regras bem definidas. Por exemplo, problemas como o da Torre de Hanói, os de criptoaritmética (aritmética cifrada) e outros semelhantes, nos quais micromundos formalizados representam os parâmetros segundo os quais os problemas podem ser solucionados. O que o GPS não conseguia fazer era solucionar problemas da vida real, como elaborar um diagnóstico médico ou tomar decisões financeiras com base em moedas de valor flutuante.

Perspectiva histórica
As pesquisas de inteligência artificial vêm se tornando cada vez mais importantes em termos práticos.
No entanto, sua história é curta em relação à de outros campos de pesquisa, como se pode ver no diagrama.

O GPS baseava-se na idéia de que a solução de qualquer problema implica uma pesquisa no espaço de suas possíveis soluções. Para torná-la eficaz, era necessário orientar tal pesquisa por regras heurísticas, ou seja, regras que permitem o aprendizado a partir de descobertas próprias, conduzindo à solução desejada por um processo de tentativa e erro. Assim, um autômato utilizaria uma técnica de busca exaustiva para sair de um labirinto se não conhecesse sua estrutura; mas, se tivesse meios de saber quando estava próximo da saída, provavelmente atingiria seu objetivo com maior rapidez (embora, às vezes, o método heurístico conduza a becos sem saída). Nesse período, várias estratégias de pesquisa heurística foram desenvolvidas pelos estudiosos da inteligência artificial.

Como o GPS não servia para a solução de problemas da vida real, uma equipe liderada por Edward Feigenbaum, da Universidade de Stanford, procurou remediar essa falha. Em vez de tentar simular por computadores a inteligência genérica, eles se concentraram, já na década de 70, em áreas bem específicas do conhecimento, desenvolvendo os sistemas especialistas.

O primeiro, o Dendral, foi elaborado em 1967 e analisava espectogramas de massa. No entanto, o mais influente foi o Mycin, apresentado em 1974, que diagnosticava infecções bacteriológicas do sangue e prescrevia drogas para o tratamento. O Mycin deu origem a toda uma família de sistemas de diagnóstico médico, alguns já em uso rotineiro, como o Puff, capaz de diagnosticar funções pulmonares, instalado no Pacific Medical Center, nos Estados Unidos.

Várias dentre as características introduzidas pelo Mycin foram adotadas por todos os sistemas especialistas. Em primeiro lugar, o "conhecimento" deles consiste em centenas de regras como a seguinte:

Regra n.º 47

SE
1) o local da cultura é o sangue, e
2) não se conhece ao certo a identidade do organismo, e
3) a coloração do organismo é gram-negativa, e
4) a morfologia do organismo é a de bastonete, e
5) o paciente sofreu queimaduras graves,

ENTÃO há uma pequena evidência (0,4) sugerindo que o organismo seja pseudomonas.

Edward Feigenbaum
Como chefe de uma equipe da Universidade de Stanford, na Califórnia, Feigenbaum criou os primeiros sistemas especialistas. Isso implicou uma reviravolta nas pesquisas, que passaram dos programas de inteligência genérica aos sistemas com objetivos e conhecimentos estritamente definidos.

Em segundo lugar, essas regras são probabilísticas. O criador do Mycin, um médico chamado Shortliffe, elaborou um esquema baseado em fatores de certeza, permitindo ao sistema tirar conclusões plausíveis a partir de dados incertos. Desse modo, o número 0,4 não é, a rigor, uma probabilidade, mas um "fator de significância". O Mycin e outros sistemas do mesmo tipo chegam a conclusões corretas mesmo partindo de informações incompletas e parcialmente errôneas. Eles conseguem esses resultados por meio de métodos de raciocínio aproximativo, como a lógica da incerteza, o cálculo de probabilidades, o recurso dos fatores de certeza etc.

Uma terceira característica é a capacidade do Mycin para explicitar seu próprio processo de raciocínio. O médico pode colocar-lhe vários tipos de pergunta com o objetivo de saber como ele chegou a uma conclusão específica ou por que requisitou determinada informação.

Por fim, o que é mais importante, o Mycin funciona. Ele faz o que requer anos de treino de um ser humano. Na verdade, o Mycin é mais empregado no ensino do que em diagnósticos, mas as grandes empresas, os órgãos governamentais e a mídia já começam a se interessar pelos princípios que ele demonstra.

Na década de 80, os sistemas especialistas ficaram na moda, graças a seu "ingrediente ativo" — o conhecimento. E a qualidade e abrangência de sua base de conhecimentos é diretamente responsável pelo sucesso de um sistema especialista. Mas o conhecimento não é algo que se possa introduzir com facilidade num computador. Codificar as habilidades de um especialista humano é um processo longo e trabalhoso. Assim, enquanto o mundo se maravilha com os sistemas especialistas, os pesquisadores da inteligência artificial dedicam-se a outro problema: o do aprendizado das máquinas, ou seja, os métodos de aquisição automática de conhecimentos. O campo da inteligência artificial sempre foi muito fluido e, no início de 1985, as pesquisas mais avançadas concentravam-se no Eurisko.

O Eurisko é um programa de investigações que amplia e melhora automaticamente, por indução, seu corpo de regras heurísticas. Vem sendo aplicado com sucesso a vários problemas práticos. Um dos resultados, na área de projetos de circuitos integrados, foi a invenção de uma inovadora porta lógica de três dimensões. Sistemas como o Eurisko constituem as pesquisas mais avançadas na área da inteligência artificial. Como esta constitui a vanguarda da ciência da computação, o que está em jogo é, nada mais nada menos, o futuro da informática.

A grande ironia na evolução resumida aqui é que, ao concentrar-se nos problemas de aprendizado, os estudiosos da inteligência artificial retornaram ao ponto de partida, pois a questão-chave nos primeiros tempos da cibernética era justamente a do aprendizado.

Nesta série de artigos sobre as áreas mais problemáticas da inteligência artificial, todos os exemplos de programas serão apresentados em BASIC. Embora a maioria das pesquisas em linguagem artificial seja realizada em LISP e PROLOG, o BASIC de seu micro servirá muito bem para a exploração inicial de algumas técnicas importantes. Afinal, um micro atual é tão poderoso quanto um computador de grande porte de vinte anos atrás. Além disso, uma vez entendidos os métodos básicos, você será capaz de implementá-los em qualquer outra linguagem.

Percepções do Perceptron
A imagem a ser analisada (a letra R) é projetada num plano e digitalizada. Os "demônios" examinam um grupo de pixels (por exemplo, quatro de cada vez) e indicam a presença ou ausência do padrão que foram programados para reconhecer. Cada resposta (expressa em dígitos zero e um) é multiplicada por um fator de ponderação, de acordo com sua importância dentro do padrão completo, e então somam-se as respostas ponderadas. O resultado é comparado a um valor limite. Se atingir o limiar, o sistema indica que o padrão é igual à forma original.

LÓGICA DA INCERTEZA

Lógica que procura reproduzir a habilidade humana de lidar com a incerteza e as meias-verdades e de tomar decisões com base em dados incompletos. Para tanto, apóia-se em valores múltiplos, possibilidades e probabilidades.

REGISTRE

Os registros são o método mais prático, no PASCAL, para agrupar diferentes tipos de dados. Antes de examinar essa nova estrutura, porém, veremos algumas operações com conjuntos.

Esta série já mostrou a utilidade dos conjuntos para o programador em PASCAL e como se revelam eficientes em nível de máquina, onde as operações por eles definidas encontram uma equivalência direta no conjunto de instruções da maioria dos processadores. Sempre construímos conjuntos reunindo um grupo de elementos de qualquer tipo escalar verdadeiro (não real). No entanto, excetuando-se a verificação da existência de um elemento por meio do operador IN, todas as operações com conjuntos são definidas na estrutura. Embora não haja mecanismo de seleção para extrair um elemento específico, empregamos os operadores de inclusão de conjunto (<= e >=) para verificar a equivalência entre conjuntos ("igualdade") com = e <>.

Quando manipulados por matemáticos, os conjuntos não apenas contêm objetos ilimitados, como também podem ser considerados teoricamente infinitos. No entanto, qualquer implementação exigirá algum tipo de limitação, tanto no tamanho como na faixa dos conjuntos. Assim, não tente definir tipos ambiciosos como:

```
NATURAIS = SET OF INTEGERS;
```

O limite inferior do conjunto deve ter um valor ordinal — de zero para cima — e o limite máximo estará restrito a algum valor absoluto — entre 255 e 4.095, nas implementações em micros. Observe que o conjunto vazio (representado literalmente por []) é membro de todos os conjuntos possíveis — qualquer que seja o tipo. Isso parece uma brecha na estrutura fortemente tipificada do PASCAL, mas, na prática, é possível deduzir o tipo do conjunto vazio a partir do tipo de outros conjuntos em qualquer expressão.

A inclusão estrita de subconjuntos verdadeiros não é direta, e tampouco se definem os operadores < e > para estruturas de conjuntos. Isso se deve a razões de implementação — a maioria das restrições do PASCAL justifica-se do ponto de vista da eficácia ou da lógica pura. Para testar a inclusão estrita é necessário um teste duplo, por exemplo:

```
(A >= B) AND (A <> B)
```

O teste para FEITO no programa de Bingo não exigia essa verificação especial, pois o conjunto dos números chamados é equivalente ("igual") ou, o mais provável, um sobreconjunto dos números na cartela. Desse modo, a expressão booleana seria expressa como

```
FEITO := CHAMADOS >= CARTELA
```

que será verdadeira quando todos os membros de CARTELA estiverem contidos no conjunto CHAMADOS. Esse tipo de propriedade faz dos conjuntos uma importante estrutura de dados para a solução de problemas. Por enquanto, o emprego mais produtivo dos conjuntos consiste no teste de subconjuntos do tipo CHAR. O esquema abaixo (em PASCAL/português) facilita a programação de qualquer software interativo:

```
NEGATIVO := ['N', 'n'];
AFIRMATIVO := ['S', 's'];

REPEAT
  {JOGAR O JOGO}
  {MOSTRAR RESULTADO}
  WRITE ('OUTRA PARTIDA?');
  READLN (RESPOSTA);

  WHILE NOT (RESPOSTA IN
    AFIRMATIVO + NEGATIVO) DO
    BEGIN
      WRITELN ('RESPONDA S(IM) OU N(AO)');
      WRITE ('OUTRA PARTIDA?');
      READLN (RESPOSTA);
    END;

UNTIL RESPOSTA IN NEGATIVO
```

Para acessar elementos individuais de uma estrutura, escolhemos entre matrizes, arquivos ou registros. Estes possuem a capacidade de reproduzir registros de dados usados na vida real, empregando "campos" mistos de qualquer tipo de dados — simples ou estruturados.

Uso de registros pelo PASCAL

Em contabilidade, a forma mais comum de registro de dados inclui campos para nomes, endereços, números de telefone, códigos de contas etc. O importante é manipular esses dados como um bloco de informações, bem como acessar qualquer campo individual e processar o dado de modo adequado. O PASCAL possibilita a atribuição (e manipulação por outros métodos) de tais objetos como uma unidade, mas também permite o acesso a qualquer campo, para comparação ou processamento, de acordo com o tipo específico de dado. A definição de um campo misto é bastante simples:

```
TYPE
  SALA = RECORD
      NUMERO : 1..999;
      POSICAO : (NORTE, LESTE, SUL,
        OESTE);
      OCUPADA : BOOLEAN
    END; {SALA}
VAR
  ESCRITORIO : SALA;
```

Assim como a instrução CASE constituía uma exceção ao uso da palavra reservada END como delimitador, a definição de um registro é a única exceção na parte declarativa à regra de que as palavras BEGIN e END devem ser utilizadas em pares. Por esse motivo, convém comentar o END de um registro com seu identificador. Qualquer variável do tipo SALA compõe-se de três campos. No exemplo, cada campo é de um tipo escalar diferente, mas eles também poderiam ser do mesmo tipo — simples ou estruturado. Não há qualquer restrição aos tipos admitidos no interior de campos de registros. Admite-se até mesmo um campo formado por um conjunto de arquivos de registros contendo conjuntos!

Entre as palavras delimitadoras RECORD e END, a sintaxe de definição é exatamente a mesma relativa à declaração VAR. Aqui, no entanto, estamos declarando identificadores de campo que fazem parte da estrutura do registro. Assim, os nomes NUMERO, POSICAO e OCUPADA não existem fora do "âmbito" do identificador do registro. Esses nomes de campo são identificadores locais e poderiam duplicar os nomes de variáveis no programa. Eles são acessados por meio dos dois mecanismos de seleção para registros do PASCAL: a notação de "ponto" e a instrução WITH.

Para selecionar determinado campo com o uso do ponto, todo o identificador do registro é separado do identificador de campo que vem a seguir por meio de um ponto ("."). Por exemplo,

```
ESCRITORIO.NUMERO
```

só se referiria ao campo de subfaixa integer. Poderíamos inicializar o conteúdo do registro com as instruções

```
READ (ESCRITORIO.NUMERO);
ESCRITORIO.POSICAO := LESTE;
ESCRITORIO.OCUPADO := PESSOAL < > [];
```

e assim por diante. Observe que usamos o conjunto vazio com um tipo que depende de PESSOAL, porém sem declará-lo.

A instrução WITH

A notação de ponto torna-se inconveniente quando queremos acessar a maioria ou todos os campos de um registro. Existe uma instrução estruturada alternativa que indica os identificadores de campo. A semântica da instrução WITH significa, de modo aproximado: "Para fazer algo com esse registro, preciso especificar somente os nomes de campo". A sintaxe da instrução WITH possui a mesma forma do loop WHILE, e a seqüência de atribuições de inicialização seria melhor expressa por:

```
WITH ESCRITORIO DO
  BEGIN
    NUMERO := 123;
    POSICAO := OESTE;
    OCUPADA := TRUE
  END
```

Em toda a instrução WITH (desde BEGIN até END, nesse caso), os identificadores de campo não precisam ser qualificados pelo identificador de registro e pelo ponto, e a referência a eles é direta. Não há confusão se outra variável do programa recebe o nome Numero, por exemplo. O âmbito local sempre tem prioridade. Todavia, usando a notação de ponto, comunicamos valores para além dos limites do âmbito. Por exemplo,

Operações com conjuntos no PASCAL

Estão relacionadas aqui todas as operações com conjuntos existentes no PASCAL (com os correspondentes diagramas de Venn, quando apropriado).

1. União (S1 + S2): sobreconjunto que abrange todos os objetos, tanto do conjunto S1 como do S2 (diagrama 1).
2. Diferença (S1 – S2): subconjunto dos membros de S1 que também não pertencem a S2 (diagrama 2).
3. Interseção (S1 * S2): subconjunto de objetos comuns a ambos os conjuntos (diagrama 3).
4. Equivalência (S1 = S2): quando os membros de S1 e S2 são idênticos (diagrama 4).
5. Não equivalência (S1 < > S2): verdadeira se nenhum dos membros de S1 for membro de S2 e vice-versa (diagrama 5).
6. Inclusão (S1 < = S2): verdadeira se todos os membros de S1 também forem membros de S2.
7. Inclusão (S1 > = S2): verdadeira se todos os membros de S2 também forem membros de S1.
8. Pertinência (m IN S1): verdadeira se o conjunto de elemento único ([m]) for um subconjunto de S1.

1

2

3

4

5

Tacada perfeita

O problema apresentado na matéria anterior, no qual pedimos o cálculo da maior seqüência de tacadas num jogo de bilhar, resolve-se assim:

```
PROGRAM    BREAKMAX    (OUTPUT);

CONST
  VERMELHAS = 15;

TYPE
  SNOOKER = (BRANCA, VERMELHA, AMARELA, VERDE,
    MARROM, AZUL, ROSA, PRETA);

VAR
  BOLA1: SNOOKER;
  PONTOS: VERMELHAS .. MAXINT;

BEGIN
  PONTOS := VERMELHAS * (ORD (VERMELHA) + ORD (PRETA));

  FOR BOLA1 := AMARELA TO PRETA DO
    PONTOS := PONTOS + ORD (BOLA1);

  WRITELN ('O BREAK MAXIMO E:', PONTOS : 3);
END.
```

```
ESCRITORIO.NUMERO := Numero
```

atribuiria um valor da variável exterior (Numero) ao campo de registro (NUMERO). O PASCAL não distingue maiúsculas e minúsculas, e o identificador não qualificado NUMERO refere-se a uma variável externa, a menos que esteja numa instrução WITH. De qualquer forma, seria essencial utilizar a notação de ponto se tivéssemos uma atribuição entre duas variáveis do mesmo tipo de registro, tal como

```
ESPERA.POSICAO := RECEPCAO.POSICAO
```

A instrução WITH é usada apenas para indicar um dos campos de variável, caso contrário haveria uma ambigüidade inaceitável para o compilador. O melhor que poderíamos fazer seria:

```
WITH ESPERA DO
  POSICAO := RECEPCAO.POSICAO
```

Portanto, as duas formas de rotação têm sua utilidade. Em geral, a forma mais adequada torna-se evidente durante a própria utilização.

Corrida de longa distância

O programa Distâncias lê dois comprimentos, expressos em jardas, pés e polegadas, com o uso do teclado. Definimos um registro a fim de reservar um campo separado para cada valor das diferentes unidades de medida. O programa soma os comprimentos, transportando as polegadas para o campo "POL" e os pés excedentes para o campo "PES", usando os operadores integer DIV e MOD para obter a soma e o resto. Os resultados são atribuídos aos campos de "TOTAL" e depois impressos. (A conversão dessas unidades de medida para o sistema métrico decimal seria a seguinte:

1 jarda = 0,9144 metro
1 pé = 0,3048 metro
1 polegada = 0,0254 metro.)

```
PROGRAM    DISTANCIAS    (INPUT, OUTPUT);
CONST
  BYTEMAXIMO = 255;

TYPE
  BYTE            = 0 .. BYTEMAXIMO;

  DISTANCIA       = RECORD
                      JAR    : 0 .. 11;
                      PES    : 1 .. 2;
                      POL    : BYTE;

                    END; DISTANCIA;
VAR
  POLEGADA;
  PE                  : BYTE;
  DISTANCIAA,
  DISTANCIAB.
  TOTAL               : DISTANCIA;

BEGIN

  WRITELN ('ENTRE DISTANCIAS COMO JARDAS, PES E
    POLEGADAS');
  WRITELN ('SEPARADOS POR ESPACO OU RETURN');
  WRITELN;
  WRITE ('PRIMEIRA :' : 15);
  READ (DISTANCIAA.JARDAS, DISTANCIAA.PES,
    DISTANCIAA.POL);
  WRITE ('SEGUNDA :' : 15);

  WITH DISTANCIAB DO
    READ (JARDAS, PES, POL);

  POLEGADA := DISTANCIAA.POL + DISTANCIAB.POL;
  PE := DISTANCIAA.PES + DISTANCIAB.PES + POLEGADA
    DIV 12;
  WRITELN;

  WITH TOTAL DO
    BEGIN
      POL := POLEGADA MOD 12;
      PES := PE MOD 3;
      JAR := DISTANCIAA.JAR + DISTANCIAB.JAR + PE
        DIM 3;
      WRITELN ('A DISTANCIA TOTAL E :');
      WRITELN (JAR : 25, 'JARDAS', PES : 1,
        'PES E', POL : 1, 'POL');
    END;
END.
```

Proteção dos dados

No programa Bingo, apresentado no artigo anterior, estabelecemos um exercício em que você devia acrescentar estruturas em loop ao programa para evitar a repetição de um número já chamado e detectar quaisquer números ilegítimos além da faixa de 1 a 90. O melhor modo de proteger o programa Bingo contra a entrada errônea de dados consiste em acrescentar uma estrutura WHILE após as duas instruções READLN. A estrutura será a mesma em cada caso:

```
WHILE o número for ilegítimo DO
  fornecer uma mensagem de erro adequada
  escrever um outro indicador
  ler os dados novamente
```

No caso das entradas iniciais de dados, devemos nos assegurar de que o número fornecido não está fora da faixa permitida (de 1 a 90), bem como evitar a repetição de uma entrada anterior. Assim:

```
WHILE NOT (NUMERO IN POSSIVEIS)
  OR (NUMERO IN CARTELA) DO
  BEGIN
    WRITELN (NUMERO : 20, 'E INVALIDO');
    WRITE ('REENTRE :' : 14);
    READLN (NUMERO);
  END;
    {ETC.}
```

A mesma estratégia se aplica à chamada dos números, mas modifica-se a condição para a entrada do loop de validação para:

```
WHILE NOT (NUMERO IN POSSIVEIS)
  OR (NUMERO IN CHAMADOS) DO
  {ETC.}
```

TRAÇOS E DÍGITOS

Apesar de muitos micros possuírem excelentes recursos gráficos, a transferência de uma foto ou desenho do papel para o vídeo revela-se demorada e difícil. A melhor solução é o traçador digital.

Um traçador digital é um equipamento simples que reproduz as linhas de uma planta, fotografia ou desenho e, ao mesmo tempo, transfere-as para a tela do computador. Trata-se, portanto, de um periférico que simplifica bastante a introdução de imagens no computador. A facilidade do processo depende sobretudo do software aplicativo que acompanha o equipamento.

O princípio de funcionamento é basicamente o mesmo em todos os modelos de traçadores digitais. Um ponteiro na extremidade de um braço articulado envia sinais elétricos ao computador. Estes variam em magnitude conforme a posição do ponteiro sobre a figura, sendo depois convertidos em sinais digitais pelo computador e usados para desenhar um ponto na tela, no lugar correto.

Todos os traçadores são comercializados com um programa aplicativo que executa essa tarefa e que dispõe de vários recursos — por exemplo, desenhar linhas em diversas cores. Outros aplicativos permitem selecionar diferentes modos de apresentação na tela, diminuindo a resolução à medida que se aumenta o número de cores, em função do limite de espaço disponível na memória.

Um traçador digital que vem obtendo êxito no mercado internacional é o Robot Plotter. Produzido pela empresa inglesa Robot Computer Development, vem montado numa prancheta transparente, onde um quadriculado representa as posições dos pixels. O braço do traçador — fixado numa extremidade da prancheta e feito em metal e plástico — é muito resistente. O ponteiro assemelha-se a uma caneta que se projeta do braço.

A figura a ser copiada fica embaixo da base transparente. Esse sistema, contudo, dificulta a visualização da figura à medida que ela vai sendo copiada.

Acompanha o Robot Plotter uma fita cassete que traz o programa aplicativo. Além das rotinas de traçado, o programa contém várias outras para desenhar círculos, retângulos e linhas.

Programas traçadores desse tipo armazenam todas as imagens como uma série de linhas; assim, o desenho de um mapa, por exemplo, é introduzido na memória como uma seqüência de linhas curtas. Isso facilita a remoção das linhas indesejáveis sem afetar outras, contíguas. Uma imagem complexa requer muito espaço de memória — às vezes, mais do que o micro dispõe.

Robot Plotter
A imagem a ser copiada é colocada sob a prancheta transparente. O software aplicativo inclui rotinas para o traçado de círculos e retângulos.

Traçador digital RD Labs
Esse modelo, de plástico e muito leve, é vendido sem a prancheta, mas pode ser utilizado em qualquer superfície, graças a seu suporte aderente.

No entanto, por serem armazenadas sob a forma de seqüências de linhas, é relativamente fácil transferir para outros programas as imagens criadas por meio do traçador.

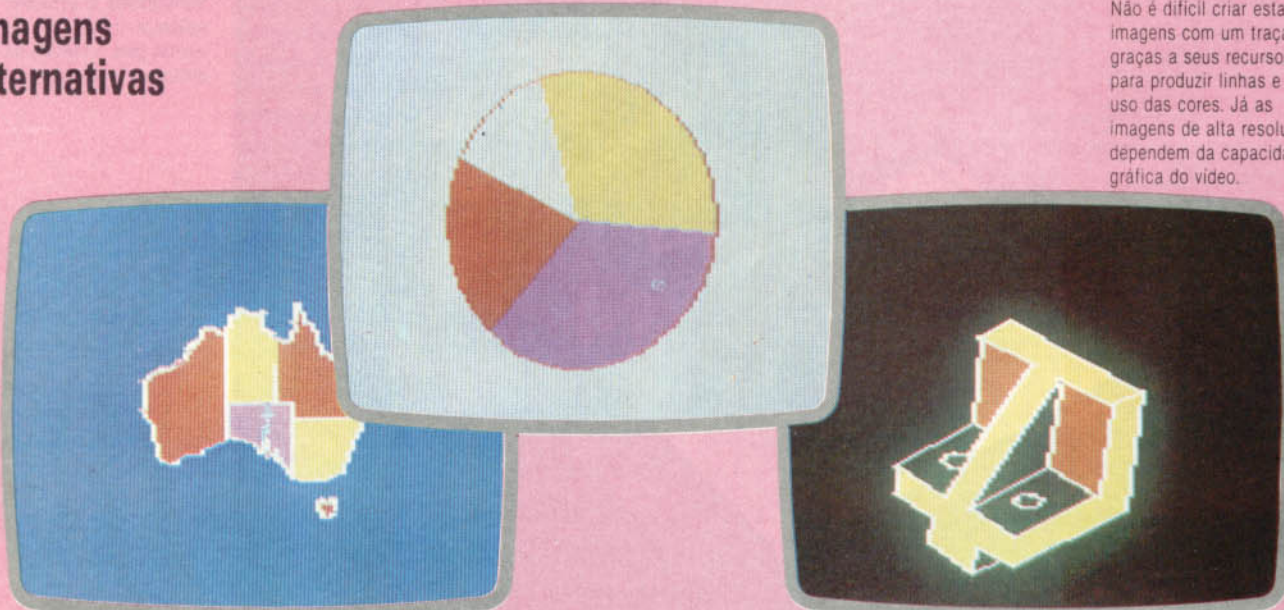
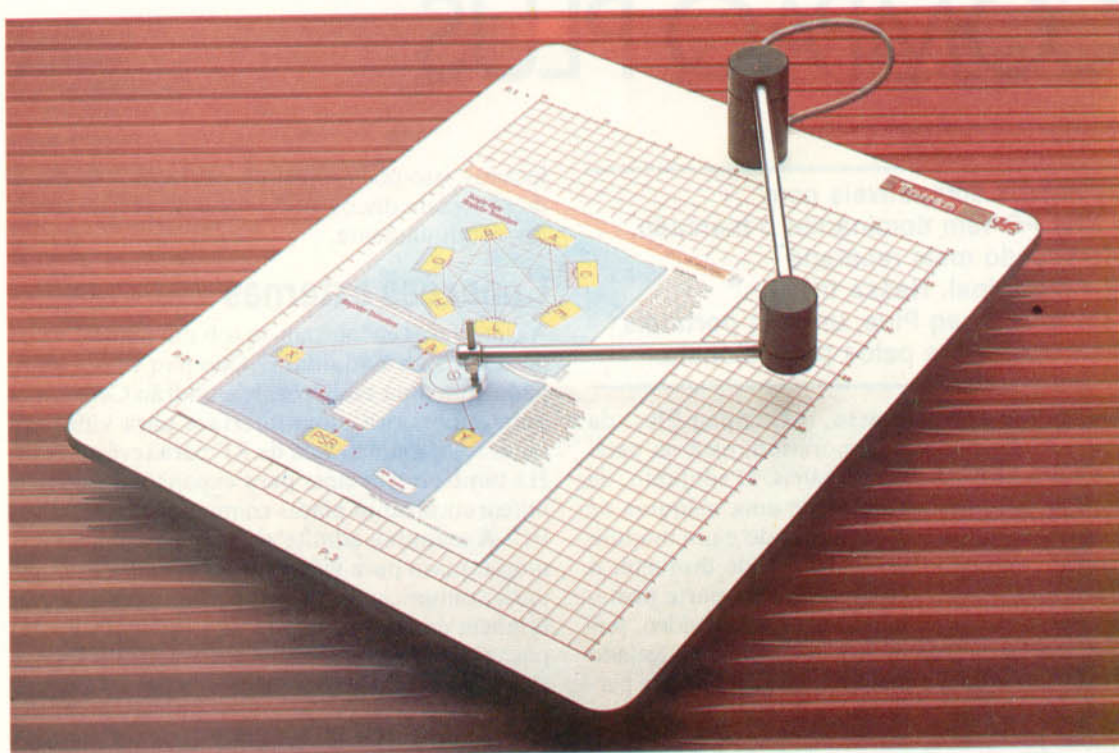
O traçador Digigraph também é bastante resistente. A placa da base consiste numa grande prancheta de madeira com um quadriculado vermelho pintado na superfície. O braço, um tubo de alumínio, vem equipado com um disco transparente na extremidade do ponteiro. Coloca-se a figura a ser copiada sobre a prancheta; o braço move-se com suavidade e a transparência do disco permite visão clara da imagem.

O aplicativo que acompanha o Digigraph, embora menos sofisticado que o do Robot Plotter, oferece recursos semelhantes. Os movimentos do traçador não se armazenam sob a forma de comandos isolados para o computador e, portanto, as figuras são transferidas diretamente para a tela. A gravação e a leitura das imagens se fazem a partir da memória de vídeo, o que torna mais difícil alterá-las. Todavia, esse processo per-

Interfaces especiais

Os fabricantes do traçador digital RD Labs desenvolveram uma interface especial para a conexão de seu equipamento ao micro Spectrum.

Imagens alternativas

Não é difícil criar estas imagens com um traçador, graças a seus recursos para produzir linhas e ao uso das cores. Já as imagens de alta resolução dependem da capacidade gráfica do vídeo.

Digigraph
Com braço de alumínio e plástico e prancheta de madeira, o traçador Digigraph é um dos mais resistentes do mercado.

mite que a memória necessária para a elaboração de uma figura complexa não seja superior à exigida por uma figura simples.

O traçador digital RD Labs é comercializado em duas versões — uma para o BBC Micro e outra para o Spectrum, da Sinclair, cujo compatível brasileiro é o TK 90X da Microdigital. Nos dois modelos, o braço vem solto, com um suporte aderente que permite sua fixação à superfície apropriada. De plástico e muito flexível, o braço opera com razoável precisão. O ponteiro utiliza dois pequenos plásticos cruzados, o que atrapalha um pouco seu uso.

Na versão para o BBC Micro, as imagens são gravadas de forma semelhante à do Robot Plotter, havendo o mesmo problema de escassez de memória. O aplicativo em cassete, bastante complexo, contém rotinas para desenho de círculos, retângulos e linhas, além de um recurso para animação de imagens. A versão para o Spectrum é basicamente a mesma, mas conta com um conversor analógico-digital — inexistente no outro modelo.

Os traçadores digitais ainda não estão sendo fabricados no Brasil. Portanto, a única alternativa para o usuário que necessite de um equipamento desse tipo é a importação e adaptação de algum dos modelos existentes no exterior. Para adequá-lo a seu micro, serão necessárias duas providências trabalhosas: modificação do conector que faz a ligação do traçador com a saída para joystick ou para paddle; e compatibilização do software aplicativo com a versão BASIC de seu aparelho.

Ângulos diferentes

Ângulo de incidência
À medida que o ponteiro se move pela prancheta, variam os ângulos no braço do traçador e os principais eixos da mesa de trabalho. Esses ângulos são traduzidos em sinais elétricos e enviados ao computador.

Ângulo de resistência
Cada articulação do traçador contém um potenciômetro, ou resistor variável. Conforme os movimentos do braço do traçador, o cursor do potenciômetro se move pela resistência, fazendo a corrente que passa variar proporcionalmente ao ângulo detectado.

Registre

Linha Apple

TELEX

Este pequeno programa em BASIC permite o movimento automático das informações contidas na tela.

```
100  REM ***  TELEX PARA LINHA APPLE ***
110  HOME
120  PRINT "ENTRE COM UM STRING E ELE PA
     SSARA EM":PRINT "SCROLL PELA SUA TE
     LA DA ESQUERDA PARA"
130  PRINT "A DIREITA":INPUT A$
140  HOME:VTAB 11
150  X=LEN(A$):IF X>39 THEN 110
160  X=40
170  HTAB 1:PRINT LEFT$(A$,X);:A$=MID$(A
     $,2)+LEFT$(A$,1)
180  FOR S=1 TO 75:NEXT S
190  K=PEEK (-16384)
200  IF K<128 THEN K=FRE(0):GOTO 170
210  END
```

COMPAQ PLUS

Os micros compatíveis com o IBM PC vêm dominando o mercado, oferecendo mais recursos que o original. Nessa trilha está o Compaq Plus, um dos portáteis mais cobiçados pelos profissionais.

Apesar dos 14 kg de peso, o Compaq Plus, da Compaq Computer Corporation, está na categoria dos profissionais portáteis. O conjunto, de dimensões aproximadas às de uma máquina de costura, compõe-se de um teclado e um módulo central contendo duas unidades de disquete, o vídeo e as placas. Há uma alça na parte posterior, e o teclado, fechando-se contra o vídeo, serve como base do conjunto. A conexão do teclado à máquina é feita por cabo espiralado, o que permite afastá-lo por cerca de 20 cm; além disso, ambos possuem ajuste de inclinação, para posicionamento no ângulo mais confortável ao operador.

Com desenho idêntico ao do IBM PC, o teclado possui dez teclas de função e dez de calculadora, além das principais, tipo QWERTY. A tecla é um monitor de fósforo verde com dimensões de 17,7 x 13,3 cm, que mostra 80 x 25 caracteres com boa nitidez e controle de luminosidade.

O Compaq Plus incorpora duas unidades de discos flexíveis de 5 1/2 polegadas. Como a configuração padrão do equipamento é de apenas uma unidade de disco e o sistema operacional MS-DOS pede que todos os discos sejam nela colocados, isso pode constituir um incômodo quando se copiam discos, pois o usuário precisa trocá-los continuamente.

Conexões externas

As interfaces encontram-se sob um painel, no lado direito da máquina. O Compaq Plus é equipado com uma saída paralela padrão Centronics para impressora, uma interface para vídeo em cores RGB e uma saída de RF para tevê comum. Há também três slots para expansões, que permitem encaixar as placas compatíveis com o IBM PC. A máquina admite ainda expansões de memória, placa para vídeo em cores ou um modem para comunicação via rede telefônica. Dispõe também de um ventilador para refrigeração das placas. Os circuitos vêm com revestimento metálico (principal responsável pelo peso), destinado a blindar o equipamento contra interferências de rádio e a dissipar o calor, além de proteger de manuseio descuidado.

O Compaq Plus, tal como o IBM PC, utiliza um processador 8088 da Intel. Alguns de seus rivais — entre eles o M25 da Olivetti — usam um chip mais avançado, o 8086. Apesar de o 8088 ser um processador de 16 bits com um bus de endereçamento de 20 bits, possui um bus de dados de 8 bits, contra o de 16 bits do 8086. Assim, embora rode na mesma velocidade que o IBM PC, o Compaq é mais lento no acesso aos dados que seus competidores equipados com 8086. O ponto alto da máquina está em ser um dos mais confiáveis compatíveis ao IBM PC à disposição no mercado internacional.

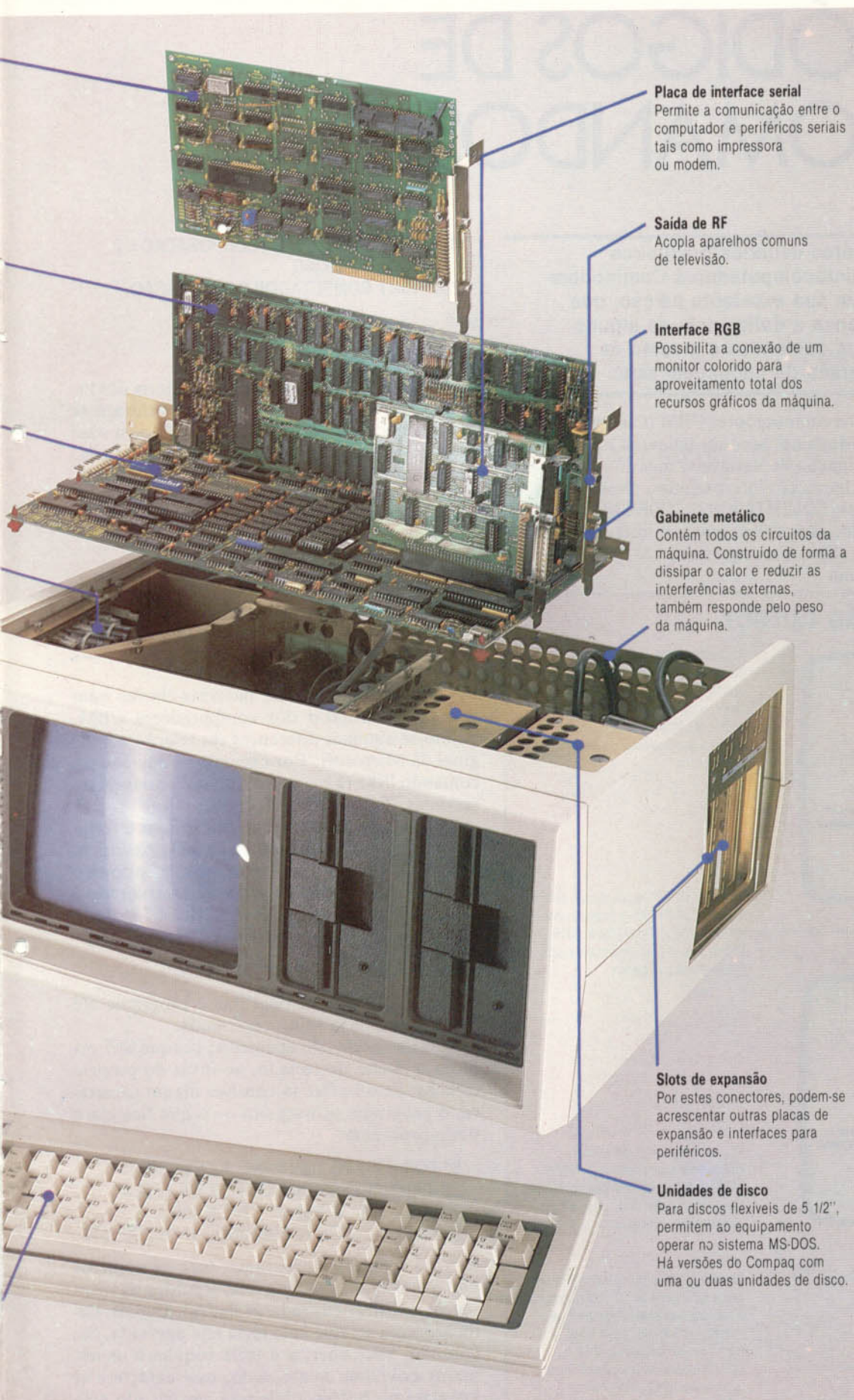
Compacto
As linhas elegantes e harmoniosas do gabinete do Compaq refletem-se no painel frontal e no teclado, este muito semelhante ao do IBM PC. Aberturas em ambos os lados dão acesso às saídas, à fonte de alimentação e ao compartimento que acomoda o fio condutor de energia — detalhe importante que costuma ser negligenciado.

Placa da impressora
Com interface padrão Centronics, permite acoplamento de uma impressora paralela.

Placa de vídeo
Encaixa-se a um dos slots de expansão e fornece os circuitos necessários para acionar um monitor ou tevê comum.

Placa principal
Contém RAM de 256 Kbytes, o processador 8088 e os chips de entrada/saída. Há também slots disponíveis para chips extras, como o processador aritmético 8087.

Fonte de alimentação
Ao lado da entrada de energia encontra-se um ventilador, que refrigera o interior do equipamento.

Teclado padrão IBM
Idêntico ao do IBM PC, até na inclinação. À esquerda estão as teclas de função e à direita as de calculadora.

Placa de interface serial
Permite a comunicação entre o computador e periféricos seriais tais como impressora ou modem.

Saída de RF
Acopla aparelhos comuns de televisão.

Interface RGB
Possibilita a conexão de um monitor colorido para aproveitamento total dos recursos gráficos da máquina.

Gabinete metálico
Contém todos os circuitos da máquina. Construído de forma a dissipar o calor e reduzir as interferências externas, também responde pelo peso da máquina.

Slots de expansão
Por estes conectores, podem-se acrescentar outras placas de expansão e interfaces para periféricos.

Unidades de disco
Para discos flexíveis de 5 1/2", permitem ao equipamento operar no sistema MS-DOS. Há versões do Compaq com uma ou duas unidades de disco.

COMPAQ PLUS

MICROPROCESSADOR
Intel 8088.

CLOCK
4,7 MHz.

MEMÓRIA
256 Kbytes de RAM, expansível até 640 Kbytes, e 8 Kbytes de ROM.

SISTEMA OPERACIONAL
MS-DOS.

VÍDEO
Monitor de fósforo verde de 17,7 x 13,3 cm.
Modo texto, 25 linhas de 80 colunas; modo gráfico, 640 x 200 pixels.

TECLADO
Com inclinação ajustável, tem 81 teclas, sendo catorze de controle, dez de funções e dez de calculadora.

LINGUAGENS
BASIC (carregado a partir do disco-mestre) e todas as que rodam sob MS-DOS, como COBOL, FORTRAN, PASCAL.

INTERFACES
Saída paralela padrão Centronics, conector de RF e RGB. Há também slots de expansão para conexão de outras placas de interfaces.

DOCUMENTAÇÃO
Acompanham três manuais: guia de operação, guia de referência do MS-DOS e guia de referência de BASIC. Dos três, apenas o guia de operações mostra passo a passo todos os procedimentos. Para os iniciantes, talvez sejam necessários outros manuais auto-instrutivos.

CÓDIGOS DE COMANDOS

Os muitos usuários brasileiros dos microcomputadores Commodore elogiam sua excelente edição, que compensa a deficiência de alguns de seus comandos e mesmo de sua versão da linguagem BASIC.

Todos os computadores CBM (Commodore Business Machine) aceitam palavras longas como denominação de variáveis, mas só as duas primeiras letras são interpretadas. Desse modo, as palavras COMPUTADOR e CORPORACAO, por exemplo, são aceitas mas consideradas equivalentes. Em conseqüência, o resultado do programa

```
100 COMPUTADOR = 17 : CORPORACAO = 2
    * COMPUTACAO
200 PRINT COMPUTADOR,CORPORACAO
```

será

```
34    34
```

Devido à forma como trabalha a maioria dos interpretadores BASIC, a velocidade se elevará se a inicialização do programa for feita com suas variáveis colocadas em ordem de importância. Consegue-se isso por meio da declaração de atribuições ou com o uso do comando DIM. Uma instrução como

```
10 DIM A$(10,24),K,L,TOTAL
```

não terá efeitos óbvios, mas é uma forma rápida de destacar as variáveis K, L e TOTAL. Faz também com que elas se tornem mais acessíveis ao interpretador; logo, incrementa a velocidade do programa.

Verificando a lista de palavras-chaves num manual do usuário dos computadores CBM, notam-se algumas diferenças em relação ao original da Microsoft. Por exemplo, a omissão do comando INKEY$ e o acréscimo das variáveis reservadas TIME$ e STATUS.

INKEY$, que tem a função de explorar o teclado, é substituído pelo comando GET. Assim como INKEY$, GET faz com que o primeiro caractere do buffer do teclado seja explorado e o transforma em seu valor ASCII. GET é usado mais freqüentemente em rotinas como

```
150 GET GT$ : IF GT$ = " "THEN 150
```

que tem a função de paralisar o programa até que alguma tecla seja pressionada.

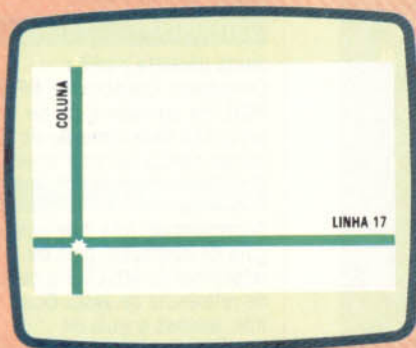
Mas isso pode não acontecer, porque GET explora o buffer do teclado, ao invés do próprio teclado. Se o buffer já contiver algum caractere, o programa prosseguirá — o que fica claro neste programa:

```
50 FOR K = 1 TO 100 : PRINT K : NEXT K
60 PRINT "PRESSIONE QUALQUER TECLA"
150 GET GT$ : IF GT$ = " "THEN 150
200 PRINT "VOCE PRESSIONOU A TECLA";GT$
```

Se você rodar esse programa, verá os números de 1 a 100 aparecerem na tela, seguidos da mensagem de INPUT (entrada de dados). Depois, nada acontece até que uma tecla seja apertada. Se, contudo, você apertar a tecla enquanto os números estiverem aparecendo, esse caractere se instalará no buffer, onde será encontrado pelo comando GET.

Posicionamento do cursor

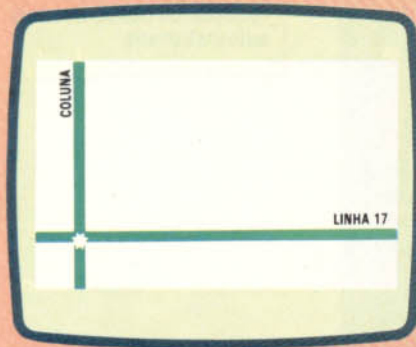
COLUNA

LINHA 17

A possibilidade de incluir os comandos de cursor numa matriz quantitativa facilita o desenho gráfico nos computadores Commodore, sobretudo quando acoplado aos poderosos comandos de edição.

```
100 PRINT AT(17,4)" * "
```

Se o BASIC da Commodore aceitasse o comando PRINT AT, o posicionamento do cursor seria simplificado. Mas essa limitação pode ser contornada pela programação do cursor.

COLUNA

LINHA 17

Inicialize igualando a variável POSICAO$ a um HOME e 24 CRSR DOWNs (cursores para baixo). Então, coloque os parâmetros da linha e da coluna nesta expressão:

```
50 POSICAO$ = "SQQQQQQQQQQQQQQQQQQQQQQQQ"
100 PRINT LEFT$(POSICAO$,17)TAB(4-1)" * "
```

Se você tiver muitas formatações de tela a fazer, será aconselhável colocar o comando de posicionamento de cursor numa sub-rotina e então inicializar as variáveis LINHA e COLUNA com as posições de tela requeridas antes de chamar a sub-rotina.

```
50 POSICAO$ = "SQQQQQQQQQQQQQQQQQQQQQQQQ"
100 LINHA = 17 : COLUNA = 4 : GOSUB 1000 : PRINT " * "
500 END
1000 PRINT LEFT$(POSICAO$,LINHA)TAB (COLUNA-1); : RETURN
```

No caso, não haverá pausa na execução do programa, o que incomoda o usuário e até impossibilita jogos em que certas teclas são mais freqüentemente acionadas. A solução está em anular ([Reset]) o buffer antes de o comando GET ser executado. Para isso, basta inserir no programa

```
149 POKE KBPTR,0
```

onde KBPTR é o endereço do contador seqüencial do teclado (198 no Commodore 64).

A variável reservada TIME$, cuja abreviatura é TI$, é o clock do sistema; quando o computador for ligado ele estará em 000000 e contará o tempo em horas, minutos e segundos até 235959 — ou seja, 23 horas, 59 minutos e 59 segundos, desde que o computador foi ligado. Então ele se anulará, voltando para o começo.

Associado a TI$ há TI, uma subdivisão numérica de TI$. TI conta o tempo em 60 avos de segundo (chamados "jiffys"). Os jiffys variam entre 0 e 5183999(60 * 60 * 60 * 24-1). TI depende inteiramente de TI$ e não pode ser inicializado por si. Apenas TI$ se inicializa.

STATUS, abreviado por ST, é uma variável definida pelo sistema. Quando se detecta um erro numa operação de entrada ou saída de dados, o valor de ST corresponde ao número que indica o tipo de erro encontrado. Normalmente, ele será escrito como

```
330 IF ST > 0 THEN GOSUB 30000 : REM E/S ERRO
    NO TRATAMENTO DE ARQUIVOS
  :
  :
30000 REM MENSAGENS DE ERRO
30100 IF ST = 16 PRINT "ERRO IRRECUPERAVEL
      DE LEITURA"
```

CMD é uma instrução muito útil dos computadores Commodore. Sua função está em direcionar o conteúdo da tela para qualquer outro canal de saída. Ela tem muitas aplicações, como

```
OPEN 4,4 : CMD 4 : LIST
```

que lista o programa na impressora, ao invés de na tela. Quando a listagem estiver completa, deverá ser executado:

```
PRINT # 4 : CLOSE 4
```

Utiliza-se CMD num programa sempre que se quer imprimir o que estiver na tela. Suponha que GOSUB 3000, em seu programa, faça com que uma mensagem ou alguns dados sejam escritos na tela. Para comandar a cópia dessa tela para a impressora, faça o seguinte:

```
OPEN 4,4 : CMD 4 : GOSUB 3000 : PRINT #
  4 : CLOSE 4
```

Embora um ponto de interrogação (?) seja aceito como abreviação do comando PRINT, não se pode abreviar o comando PRINT# por ?# e sim por pR.

Na prática, qualquer comando dos computadores Commodore pode ser abreviado, usando-se para isso sua primeira letra em minúscula seguida pela segunda em maiúscula. Quando dois ou mais comandos têm as duas primeiras letras iguais, a abreviação deve ser feita com as duas primeiras letras em minúsculas, seguidas da terceira, em maiúscula. Por exemplo, READ, RESTORE e RETURN abreviam-se rE, reS e reT.

A edição, reforçada pelo sistema operacional facilitado ao usuário, é a mais significativa característica dos computadores Commodore. Vem sendo usada desde o lançamento do PET e é até hoje considerada a melhor de todas.

Para editar uma linha do programa, basta listá-la, movimentar o cursor para qualquer ponto da linha, editar o texto e pressionar a tecla [Return]. Independentemente da localização do texto na tela ou do cursor na linha, quando [Return] é pressionada, o texto contido na linha em que se localiza o cursor entra no sistema como se tivesse acabado de ser digitado.

Uma vantagem dessa edição está na facilidade de reprodução. Suponha a necessidade de digitar estas duas linhas:

```
100 IF INT(NUMERO/INDICE—TAXA) = 5 THEN
    3000
200 IF INT(NUMERO/INDICE—TAXA) = 7 THEN
    3800
```

Digite a primeira linha e acione a tecla [Return]; então, mantendo esse texto na tela, mova o cursor para cima, modifique o número da linha 100 para 200, troque o 5 por 7, troque 3000 por 3800 e pressione de novo [Return]. A linha 100 estará guardada na memória e as transformações de edição farão a linha 200 surgir na tela.

Esse processo se repete quantas vezes necessário. Trata-se apenas de um dos truques permitidos pelos recursos de edição dos computadores Commodore, mas já demonstra como seu uso é fácil e direto.

Telas versáteis
Um dos pontos altos dos equipamentos Commodore está nos seus recursos gráficos, que permitem a execução de avançados softwares para empresas, como planilhas, processadores de texto e bancos de dados.

A REVOLUÇÃO CUBISTA

Os princípios geométricos usados no programa para traçar figuras tridimensionais (projeções em perspectiva) fornecem uma base para que se desenvolvam alguns gráficos animados simples.

O programa aqui apresentado utiliza princípios geométricos básicos para criar projeções em perspectiva de objetos que podem ser observados de qualquer ângulo ou distância.

Todos os dados dos objetos são armazenados no programa por meio de instruções DATA. Essas instruções contêm as coordenadas tridimensionais de cada vértice do objeto. Para cada ponto há também um número que esclarece se ele está no início de uma linha ou em seu final (o que indica se a linha será traçada a partir do ponto ou até ele).

Converter essas coordenadas tridimensionais em valores bidimensionais representando pontos na tela é questão de simples porém extenso trabalho matemático. As coordenadas x e y de cada ponto (no plano da tela do vídeo) são divididas por um fator que representa a distância do objeto em relação ao observador. Além disso, aplica-se à coordenada resultante um fator de escala apropriado ao sistema de coordenadas do micro. A modificação do fator de distância faz o objeto diminuir ou aumentar de tamanho, conforme se aproxime ou se afaste do observador.

Usa-se, ainda, uma terceira constante para modificar o efeito da projeção em perspectiva. Aumentando o valor dessa constante, a imagem em perspectiva do objeto é exagerada, como se ele estivesse sendo observado por meio de lente grande-angular. Diminuindo a constante, obtém-se o efeito de achatamento da imagem, como se estivesse sendo observada por meio de teleobjetiva.

Além de criar uma imagem em perspectiva do objeto, o programa possibilita girá-la de modo que possa ser observada de qualquer ângulo. Isso se realiza por meio de trigonometria simples. Os eixos sofrem uma rotação até o ângulo desejado, de modo que, quando se faz a projeção em perspectiva, a imagem na tela parece ter girado. Consegue-se esse efeito com uma rotação em torno do eixo y ou do eixo x. No primeiro caso, o ponto de observação parece girar na horizontal em torno do objeto; no segundo, o ponto de observação se coloca acima ou abaixo do objeto.

Embora crie todos esses efeitos, nosso programa é extremamente simples. O controle do ponto de observação se faz pelas teclas numéricas de 1 a 8. Essas teclas darão, respectivamente: um movimento do ponto de observação para a esquerda (tecla [1]), para a direita (tecla [2]), para cima (tecla [3]), para baixo (tecla [4]), em direção ao objeto (tecla [5]) ou afastando-se dele (tecla [6]), um aumento (tecla [7]) no efeito de perspectiva (olho-de-peixe) ou uma diminuição (tecla [8]) no efeito de perspectiva (teleobjetiva).

As coordenadas tridimensionais dos pontos (vértices) são armazenadas numa única matriz: V(6,50).

A alteração dessa matriz e a alteração das constantes usadas na transformação da perspectiva são feitas por meio de uma série de sub-rotinas. Cada vez que se pressiona uma tecla, a imagem na tela é apagada por um comando HGR2 (no Apple) ou CLS (no TK 90X); a alteração nas coordenadas dos vértices se realiza por uma sub-rotina apropriada; e a nova imagem é, então, desenhada.

A transformação da perspectiva ocorre na sub-rotina que transforma a imagem. Essa sub-rotina percorre cada conjunto de coordenadas tridimensionais, transforma-as em coordenadas bidimensionais e as traça na tela (movendo-se até os vértices ou desenhando linhas, de acordo com um parâmetro adicional introduzido junto com cada coordenada tridimensional).

Os dados são armazenados em instruções DATA no final do programa, com quatro valores para cada vértice do objeto. O número total de vértices é estabelecido na primeira linha do programa (atributo da variável N). Isso deve ser modificado para adaptar-se a seus dados. Os dados que fornecemos criam um cubo com uma linha diagonal atravessando uma das faces.

Cada conjunto de quatro valores está nesta ordem: parâmetro para desenhar ponto ou traçar linha, coordenada x, coordenada y, coordenada z. Os valores são calculados traçando-se mentalmente o contorno do objeto em três dimensões. Com o auxílio de uma caneta imaginária, vá até cada vértice do contorno do objeto. Se sua caneta for movida até um vértice sem desenhar uma linha, o primeiro valor será 4. Se o primeiro valor for 5, será desenhada uma linha do vértice precedente até esse vértice.

A origem das coordenadas (0,0) está no centro da tela. É melhor fazer dela o centro do objeto. O eixo x é o eixo horizontal, com os valores positivos aumentando para a direita. O eixo y é o vertical, com os valores positivos aumentan-

do de baixo para cima. O eixo z é o que vai de fora para dentro da tela. O sentido positivo desse eixo também vai para a tela.

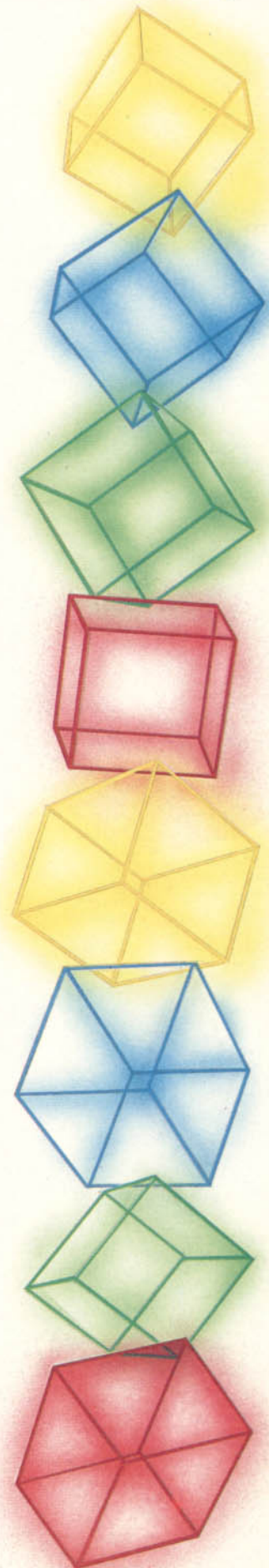
Procure usar os menores valores possíveis para as coordenadas X, Y e Z dos vértices de seu objeto. O ajuste inicial do efeito de perspectiva e de distância do ponto de observação deve levar em conta que uma largura de objeto por volta de 10 irá preencher toda a tela. Isso pode ser modificado pela alteração do fator de escala usado na transformação da perspectiva.

Faça primeiramente experiências com as formas mais simples. Procure usar poucos pontos. Quando tiver dominado a construção de objetos sólidos simples (pirâmides, cubos etc.), você passará para outros, mais complexos.

Versão Apple e TK 2000

```
10 N = 16: DIM V(6,50):D = 10:P =
      0.5:SI = SIN (0.09):CO = COS
      (0.09): HGR2
20  FOR I = 1 TO N: READ V(1,I),V
      (2,I),V(3,I),V(4,I): NEXT I
30  HCOLOR= 3: GOSUB 300
40  GET I$: HGR2
50  ON VAL (I$) GOSUB 1000,2000,
      3000,4000,5000,6000,7000,800
      0
60  GOTO 30
300  FOR I = 1 TO N:V(5,I) = V(2,
      I) * 300 / (P * V(4,I) + D):
      V(6,I) = V(3,I) * 300 / (P *
      V(4,I) + D): NEXT I
310  FOR I = 1 TO N
320  IF V(1,I) = 4 THEN HPLOT (V
      (5,I) + 140),(V(6,I) + 95)
330  IF V(1,I) = 5 THEN HPLOT (V
      (5,I - 1) + 140),(V(6,I - 1)
      + 95) TO (V(5,I) + 140),(V(
      6,I) + 95)
340  NEXT I: RETURN
1000  FOR I = 1 TO N:X = V(2,I) *
      CO - V(4,I) * SI:Z = V(4,I) *
      CO + V(2,I) * SI:V(2,I) = X:
      V(4,I) = Z: NEXT I: RETURN
1010  PRINT CHR$ (9);"40N"
1020  LIST 10,9999
1030  PRINT CHR$ (9);"PR#0"
2000  FOR I = 1 TO N:X = V(2,I) *
      CO + V(4,I) * SI:Z = V(4,I) *
      CO - V(2,I) * SI:V(2,I) = X:
      V(4,I) = Z: NEXT I: RETURN
3000  FOR I = 1 TO N:Y = V(3,I) *
      CO + V(4,I) * SI:Z = V(4,I) *
      CO - V(3,I) * SI:V(3,I) = Y:
      V(4,I) = Z: NEXT I: RETURN
4000  FOR I = 1 TO N:Y = V(3,I) *
      CO - V(4,I) * SI:Z = V(4,I) *
      CO + V(3,I) * SI:V(3,I) = Y:
      V(4,I) = Z: NEXT I: RETURN
5000 D = D * 0.9: RETURN
6000 D = D / 0.9: RETURN
7000 P = P / 0.9: RETURN
8000 P = P * 0.9: RETURN
9000  DATA 4,1,1,1,5,1,1,-1,5,-1
      ,1,-1,5,-1,1,1,5,1,1,1
9010  DATA 5,1,-1,1,5,1,-1,-1,5,
      -1,-1,-1,5,-1,-1,1,5,1,-1,1
9020  DATA 4,1,-1,-1,5,1,1,-1,5,
      -1,-1,-1,5,-1,1,-1
9030  DATA 4,-1,1,1,5,-1;-1,1
```

Versão TK 90X

```
  10 LET N=16: DIM V(6,50)
  20 LET D=10: LET P=0.5
  30 LET SI=SIN 0.09: LET CO=COS
 0.09
  40 FOR I=1 TO N
  50 READ V(1,I),V(2,I),V(3,I),V
(4,I)
  60 NEXT I
  70 INVERSE 0: GOSUB 300
  80 IF INKEY$<>"" THEN GOTO 80
  90 IF INKEY$="" THEN GOTO 90
 100 LET I$=INKEY$
 110 CLS
 120 GOSUB VAL I$*1000
 130 GOTO 70
 300 FOR I=1 TO N: LET V(5,I)=V(
2,I)*300/(P*V(4,I)+D): LET V(6,I
)=V(3,I)*300/(P*V(4,I)+D): NEXT
I
 310 FOR I=1 TO N
 320 IF V(1,I)=4 THEN PLOT V(5,I
)+127,V(6,I)+87
 330 IF V(1,I)=5 THEN DRAW V(5,I
)-V(5,I-1),V(6,I)-V(6,I-1)
 340 NEXT I: RETURN
1000 FOR I=1 TO N: LET X=V(2,I)*
CO-V(4,I)*SI: LET Z=V(4,I)*CO+V(
2,I)*SI: LET V(2,I)=X: LET V(4,I
)=Z: NEXT I: RETURN
2000 FOR I=1 TO N: LET X=V(2,I)*
CO+V(4,I)*SI: LET Z=V(4,I)*CO-V(
2,I)*SI: LET V(2,I)=X: LET V(4,I
)=Z: NEXT I: RETURN
3000 FOR I=1 TO N: LET Y=V(3,I)*
CO+V(4,I)*SI: LET Z=V(4,I)*CO-V(
3,I)*SI: LET V(3,I)=Y: LET V(4,I
)=Z: NEXT I: RETURN
4000 FOR I=1 TO N: LET Y=V(3,I)*
CO-V(4,I)*SI: LET Z=V(4,I)*CO+V(
3,I)*SI: LET V(3,I)=Y: LET V(4,I
)=Z: NEXT I: RETURN
5000 LET D=D*0.9: RETURN
6000 LET D=D/0.9: RETURN
7000 LET P=P/0.9: RETURN
8000 LET P=P*0.9: RETURN
9000 DATA 4,1,1,1,5,1,1,-1,5,-1,
1,-1,5,-1,1,1,5,1,1,1
9010 DATA 5,1,-1,1,5,1,-1,-1,5,-
1,-1,-1,5,-1,-1,1,5,1,-1,1
9020 DATA 4,1,-1,-1,5,1,1,-1,5,-
1,-1,-1,5,-1,1,-1
9030 DATA 4,-1,1,1,5,-1,-1,1
```

MÉTODOS ALTERNATIVOS

Embora os três tipos básicos de porta (E, OU e NÃO) resolvam qualquer problema lógico, as portas lógicas $\overline{\text{E}}$ e $\overline{\text{OU}}$ oferecem métodos alternativos que simplificam o desenho de circuitos.

Se podemos resolver todos os problemas lógicos com o uso de portas E, OU e NÃO, por que aprender outros tipos de porta? O motivo é a redução do custo de fabricação do circuito que essas portas permitem, quer isoladamente, quer combinadas com outras. Isso porque elas simplificam o sistema de conexão ou oferecem soluções mais refinadas. Todos os problemas lógicos podem ser resolvidos pelo uso de uma das seguintes técnicas:

a) portas E, OU e NÃO juntas;
b) somente portas $\overline{\text{E}}$;
c) somente portas $\overline{\text{OU}}$;
d) uma combinação de todas.

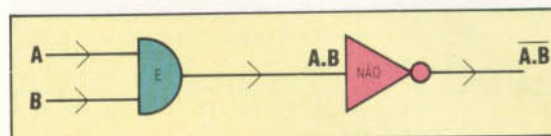
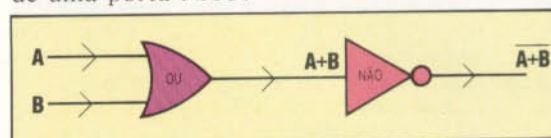
A função de cada porta é melhor descrita por sua tabela de validação:

A	B	C	PORTA $\overline{\text{E}}$
0	0	1	
0	1	1	
1	0	1	
1	1	0	

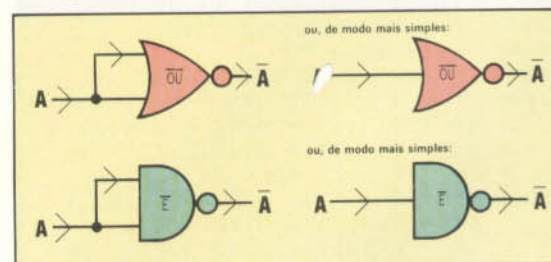
$\overline{\text{E}}$ é a representação de NÃO E, e a comparação dessa tabela de validação com a tabela usada para a porta E permite notar que na coluna da saída todos os 1 foram substituídos por 0, e vice-versa.

A	B	C	PORTA $\overline{\text{OU}}$
0	0	1	
0	1	0	
1	0	0	
1	1	0	

Do mesmo modo, $\overline{\text{OU}}$ é a representação de NÃO OU, e a comparação das colunas de saída (coluna C) para essa tabela e para a tabela da porta OU mostra que todos os 1 e 0 foram negados.

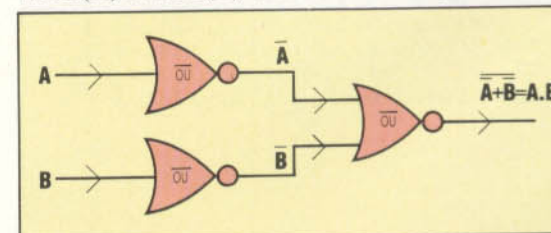
Não há símbolos especiais para as operações $\overline{\text{E}}$ e $\overline{\text{OU}}$ na álgebra booleana, mas é possível representar cada função booleana com os símbolos E, OU e NÃO, examinados anteriormente. Uma porta $\overline{\text{E}}$ equivale a este circuito simples:

E a porta $\overline{\text{OU}}$ equivale a uma porta OU seguida de uma porta NÃO:

O uso de $\overline{\text{E}}$ e $\overline{\text{OU}}$

Assim como podemos desenhar circuitos E/OU/NÃO que equivalem a $\overline{\text{E}}$ e $\overline{\text{OU}}$, também é possível representarmos cada uma dessas três portas básicas em termos de uma série de portas $\overline{\text{OU}}$ ou de uma série de portas $\overline{\text{E}}$.

Portas NÃO: A negação pode ser obtida conectando-se ambas as entradas, com o uso de uma porta $\overline{\text{OU}}$ ou de uma porta $\overline{\text{E}}$:

Portas E: Em termos de álgebra booleana, a saída de uma porta E com entradas A e B será A . B. No entanto, essa expressão pode ser convertida numa forma mais útil:

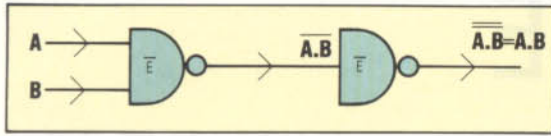
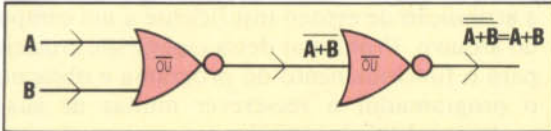
$$A . B = \overline{\overline{A}} . \overline{\overline{B}} \quad \text{(pois } A = \overline{\overline{A}}\text{)}$$

$$= \overline{\overline{A} + \overline{B}} \quad \text{(lei de De Morgan)}$$

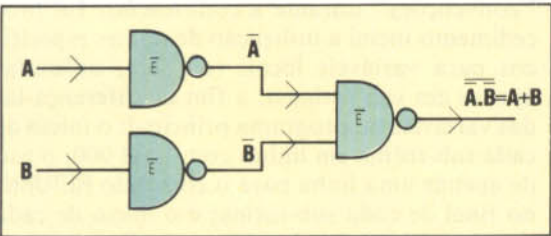
Assim, desenhamos o circuito colocando NÃO(A) e NÃO(B) diante de uma porta $\overline{\text{OU}}$:

Também podemos criar uma porta E usando portas $\overline{\text{E}}$. A saída de uma porta $\overline{\text{E}}$ é $\overline{A.B}$. Se es-

sa saída for negada, obteremos

$\overline{\overline{A.B}} = A.B$

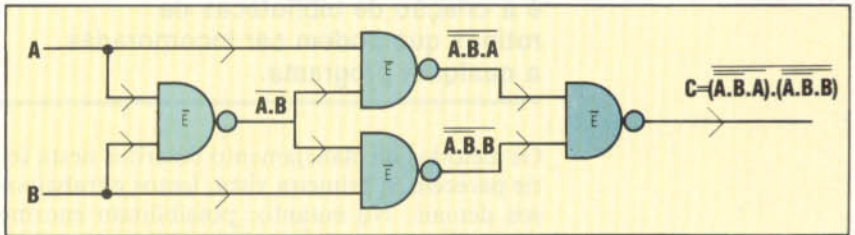
Assim, o circuito será:

Portas OU: Assim como o encadeamento de duas portas $\overline{E}$ equivale a uma porta E, do encadeamento de duas portas $\overline{OU}$ resultará um circuito equivalente a uma porta OU:

A saída para uma porta OU é A + B. Usando as regras da álgebra booleana, modificamos essa saída para uma forma $\overline{E}$:

$$A+B = \overline{\overline{A}} + \overline{\overline{B}}$$
$$= \overline{\overline{A}.\overline{B}}$$

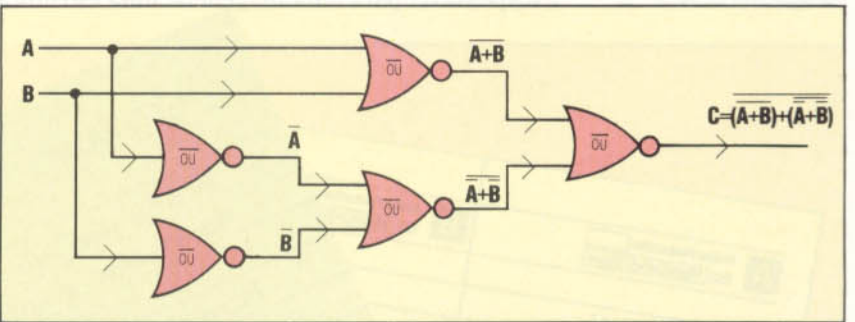
Assim, o circuito correspondente que usa as portas $\overline{E}$ será:

Para obter um circuito que utiliza apenas elementos $\overline{E}$ ou $\overline{OU}$, podem-se ainda seguir os métodos de simplificação já vistos, mas antes é preciso modificar a expressão booleana final a uma forma apropriada. Para circuitos que incorporam as portas $\overline{E}$, utilizam-se as regras da álgebra booleana a fim de criar uma expressão que consista em grupos de Es ligados por meio de OUs; e usa-se o teorema de De Morgan seguidas vezes até que a expressão esteja completamente na forma $\overline{E}$. Para circuitos da forma $\overline{OU}$, empregam-se regras semelhantes. Em demonstrar como essas regras são usadas, convém examinar a porta OU exclusivo (XOU). A saída dessa porta pode ser definida pela expressão $C = \overline{A.B}.(A+B)$.

É possível converter essa expressão de modo a constituir o circuito para a porta XOU apenas por meio de portas $\overline{E}$. Em primeiro lugar, obtêm-se grupos de Es conectados por meio de OUs:

$$C = \overline{A.B}.(A+B)$$
$$= (\overline{A.B}.A) + (\overline{A.B}.B) \quad \text{(multiplicação dos parênteses)}$$
$$= \overline{\overline{(\overline{A.B}.A)}.\overline{(\overline{A.B}.B)}} \quad \text{(teorema de De Morgan)}$$

Ao desenhar o circuito para uma expressão complexa como essa, é melhor iniciar a partir da saída e trabalhar voltando em direção às entradas. Tente seguir com essa orientação o diagrama de circuito abaixo para ver como ele foi construído.

Para a forma OU, é necessário mais uma vez começar com a expressão simplificada original para a porta XOU e modificá-la para formar grupos de OUs conectados por meio de Es. Esse primeiro passo pode ser feito com o teorema de De Morgan na parte da esquerda:

$$C = \overline{A.B}.(A+B)$$
$$= (\overline{A}+\overline{B}).(A+B)$$
$$= \overline{\overline{(\overline{A}+\overline{B})}+\overline{(A+B)}}$$

Também nesse caso a conversão da expressão num diagrama de circuitos é mais fácil se realizada a partir da saída em direção às entradas.

Respostas do exercício anterior

1a)

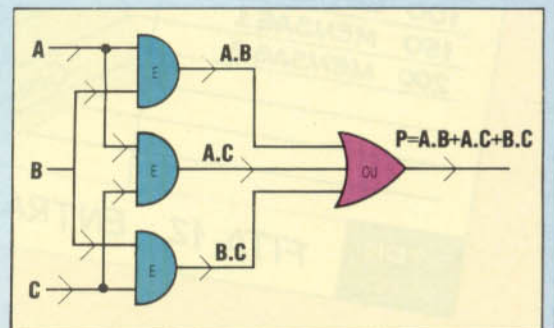
Entradas			Saída
A	**B**	**C**	**P**
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

b) A expressão booleana para P é:

$$P = \overline{A}.B.C + A.\overline{B}.C + A.B.\overline{C} + A.B.C$$

Podemos simplificar com o uso de um mapa-k:

Assim, $P = A.B + A.C + B.C$

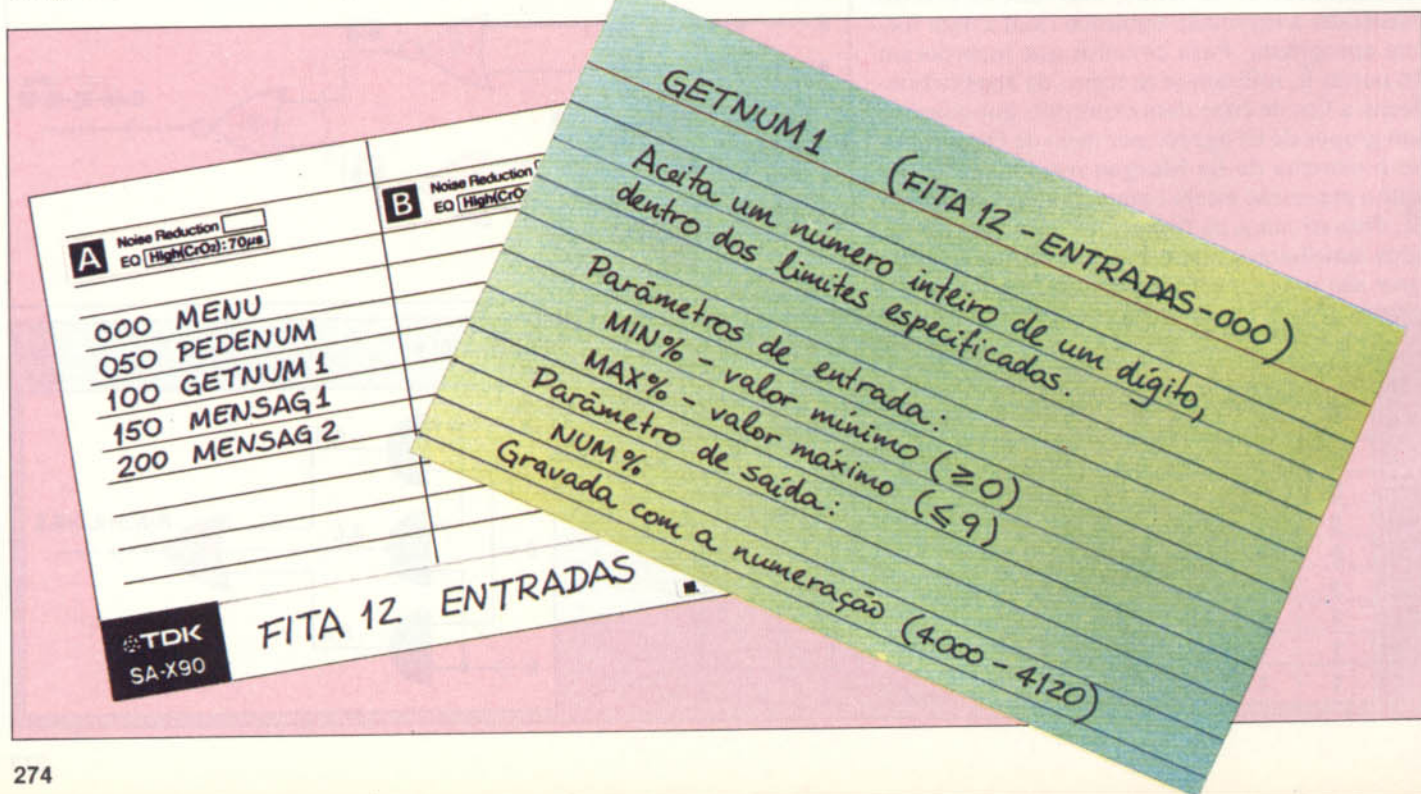
c) O circuito é:

BIBLIOTECA CIRCULANTE

De enorme utilidade são os métodos para economizar tempo e esforço em programação. Uma dessas técnicas é a criação de bibliotecas de rotinas que podem ser incorporadas a qualquer programa.

Os métodos de planejamento descritos nesta série parecem, à primeira vista, lentos e trabalhosos demais. No entanto, possibilitam enorme economia de tempo, não apenas durante a codificação, mas também na depuração dos erros. Em geral, os programas criados no teclado, sem nenhum planejamento, possuem estruturas e algoritmos desnecessariamente complicados. Isso significa que exigem muito tempo para ser desenvolvidos, estão mais sujeitos a erros e, por fim, acarretam mais trabalho quando testados e corrigidos, em razão de sua lógica defeituosa. O planejamento prévio de um programa simplifica bastante sua estrutura e seus algoritmos, diminuindo os erros de codificação e facilitando os testes e a depuração.

Mais importante ainda, o planejamento é indispensável para a montagem de uma estrutura de arquivo ou de controle que não se revele depois inteiramente inadequada — por exemplo, a atribuição de espaço insuficiente a um campo do arquivo. Problemas dessa espécie são cruciais para o funcionamento do programa e obrigam o programador a reescrever muitas de suas partes.

Aperfeiçoando a datilografia, os usuários que possuem micros com teclado profissional, semelhante ao das máquinas de escrever, são capazes de aumentar a velocidade dos toques. Fora isso, não existe nenhum outro método para tornar mais rápida a digitação das linhas do programa. Mas é possível acelerar bastante o processo de codificação dos programas. O recurso mais simples é o de inventar, adotar e usar inúmeras ''convenções'' durante a codificação. Tal procedimento inclui a utilização de nomes específicos para variáveis locais (ou seja, utilizadas apenas em sub-rotinas), a fim de diferençá-las das variáveis do programa principal; o início de cada sub-rotina em linhas com final 000; o uso de apenas uma linha para o comando RETURN, no final de cada sub-rotina; e o início de cada tipo de sub-rotina num grupo específico de linhas (manipulação de arquivos entre 9000 e 9999, utilitários a partir de 50000 etc.).

Sistema uniforme
De nada vale montar uma biblioteca de sub-rotinas sem um sistema de documentação uniforme. Isso se aplica com mais força aos usuários de fita cassete — é muito irritante ter de descobrir o conteúdo de fitas sem indicação alguma, carregando-as no micro.

O emprego dessas convenções apresenta muitas vantagens. Não é necessário procurar laboriosamente determinada rotina na listagem, pois cada tipo está sempre no mesmo lugar; não é preciso preocupar-se com a atribuição do mesmo nome para uma variável no programa principal e numa sub-rotina, porque o nome indicará se a variável é local ou global.

Biblioteca de programas

Essas técnicas de codificação também facilitam bastante a organização de uma biblioteca de programas. Uma bem organizada biblioteca de sub-rotinas permite reduzir pela metade o tempo de codificação de um programa extenso.

A melhor maneira de começar uma biblioteca é examinar todos os programas disponíveis e, então, selecionar as sub-rotinas bem escritas e de emprego geral (rotinas de entrada e saída, de conversão de maiúsculas em minúsculas etc.). Cada rotina deve ser gravada como um arquivo separado. Estes serão agrupados conforme a função (se forem armazenados em fita, cada grupo de funções é gravado em fita separada), recebendo nomes característicos para melhor identificação. Mantenha um índice ou um banco de dados com os nomes dos arquivos e a descrição de cada rotina.

Tenha certeza de que todas as rotinas da biblioteca foram testadas e depuradas. Certifique-se de que rejeitarão quaisquer valores de entrada inadequados, pois serão usadas em programas para os quais não foram especialmente criadas. Verifique também se os valores de saída causarão problemas no programa que os utilizar. Faça com que cada rotina seja o mais eficiente possível e inclua algumas linhas com comentários para facilitar sua compreensão, posteriormente. Aumente sua coleção apenas o necessário e não se esqueça de numerar as linhas das rotinas de sua biblioteca conforme as convenções estabelecidas (evitando assim renumerá-las quando forem usadas num novo programa).

Para formar o programa completo é preciso um meio de agrupar várias rotinas. Quando se usa linguagem compilada, os vários módulos são unidos com auxílio de um programa tipo link-loader, tornando-os um único programa executável. Os programadores que não dispõem de um compilador de linguagem devem utilizar-se dos comandos MERGE e RENUM, já incorporados a algumas versões do BASIC ou disponíveis no mercado como utilitários.

Quando quiser acrescentar uma rotina ao novo programa, você precisa carregar o programa, decidir onde irá introduzir a rotina e certificar-se de que existe para isso um número suficiente de linhas não utilizadas. Se necessário, renumere a rotina, para que ela ocupe o espaço que lhe cabe. Use então o comando MERGE para unir os dois programas. Confira se tudo está funcionando direito e grave o novo programa com a rotina de biblioteca já incorporada.

MERGE

- O TK 90X possui o comando MERGE incorporado a seu BASIC. Com o programa a ser completado já na memória do micro, comanda-se MERGE "nome da rotina a ser carregada" e introduz-se o programa complementar do gravador. Caso haja sobreposição de linhas, aquela que está sendo carregada prevalece sobre a outra, que é anulada.
- O Apple, o TK 85, o CP 2000 e alguns outros micros, que não possuem um comando MERGE residente, dispõem de utilitários com rotinas MERGE, RENUM e outras, em discos ou fitas. O RENUM também está disponível na forma de utilitário para o TK 90X.

Trabalho em grupo

É comum que um grupo de programadores trabalhe em conjunto num projeto — seja numa empresa, seja numa escola ou num clube de micro. A maior parte do que foi dito sobre planejamento de programas e sobre a eficiência do programador é de especial importância para o trabalho em grupo. Muitos desses conceitos e idéias de programação estruturada foram desenvolvidos para que se dividisse a tarefa de produção de programas. Isso permite aos programadores trabalharem simultaneamente em diferentes partes do mesmo programa, terminando-o com mais rapidez.

Para que programadores em BASIC possam trabalhar em grupo, é essencial que estejam de acordo quanto às convenções que serão utilizadas na codificação. Uma vez estabelecido o plano geral do programa, o programador de cada módulo precisa saber o seguinte:

1) Em que consistem os arquivos e como serão organizados.
2) Quais as convenções utilizadas para dar nome às variáveis. As variáveis mais importantes — como as matrizes, que serão utilizadas no programa todo — têm seus nomes previamente definidos. É preciso estabelecer também uma convenção para variáveis locais e denominar com antecedência as variáveis que passam de um módulo para outro, ou achar um meio de garantir que cada uma delas seja única. Para isso, adiciona-se, por exemplo, o número do módulo em que ela é definida ao seu final.
3) Quais as rotinas disponíveis para o grupo, a função e o formato de cada uma, o nome de suas variáveis e o modo pelo qual foram testadas ou depuradas.
4) De que modo estão organizadas as rotinas de controle de erros; se cada rotina corrige seus próprios erros ou apenas os indica para que sejam corrigidos pela rotina de controle do programa mais abrangente.
5) A função exata de cada módulo.
6) O tipo de dado e a faixa de valores que cada módulo individual deve aceitar como entrada e devolver como saída.

ELETRÔNICA POPULAR

Adaptando micros ingleses versáteis e de baixo custo, a Microdigital firmou-se na produção de equipamentos de consumo basicamente popular e garantiu a liderança do mercado nacional.

Com mais de 100.000 unidades vendidas no Brasil desde sua fundação, em 1981, a Microdigital Eletrônica é o maior fabricante nacional de microcomputadores, responsável por 60% do mercado. Investindo 5% de seu faturamento anual (10 milhões de dólares em 1984) em pesquisa e desenvolvimento, a empresa contava, no início de 1985, com quatrocentos funcionários, três unidades industriais (duas em São Paulo e uma em Manaus), seis unidades de serviços técnicos e mais de setecentos revendedores. No mercado internacional, suas vendas atingiram 1,5 milhão de dólares em 1984, a maior parte para a Argentina, onde estabeleceu mais de trezentos pontos de venda e comercializou 10.000 aparelhos.

Os produtos desenvolvidos pela Microdigital têm grande aceitação entre estudantes e profissionais liberais.

A Microdigital inaugurou sua produção com o modelo TK 82, seguido pouco depois por uma versão com design reformulado, o TK 82C, ainda em 1981. Desenvolvido a partir do micro ZX 80, da Sinclair, esse equipamento foi substituído, no final de 1982, pelo TK 83. Com 8 Kbytes de ROM e programável em linguagem BASIC ou ASSEMBLY, o TK 83 vem equipado com teclado de membrana e duas interfaces para impressora e monitor de vídeo. De custo menor, esse micro teve larga penetração junto ao público iniciante na informática.

Dispondo de mais recursos que os modelos anteriores, o TK 85 chegou ao mercado em 1983, com ROM de 10 Kbytes. Seu teclado emborrachado, tipo chiclete, compõe-se de quarenta teclas no padrão QWERTY e permite acesso a 160 funções.

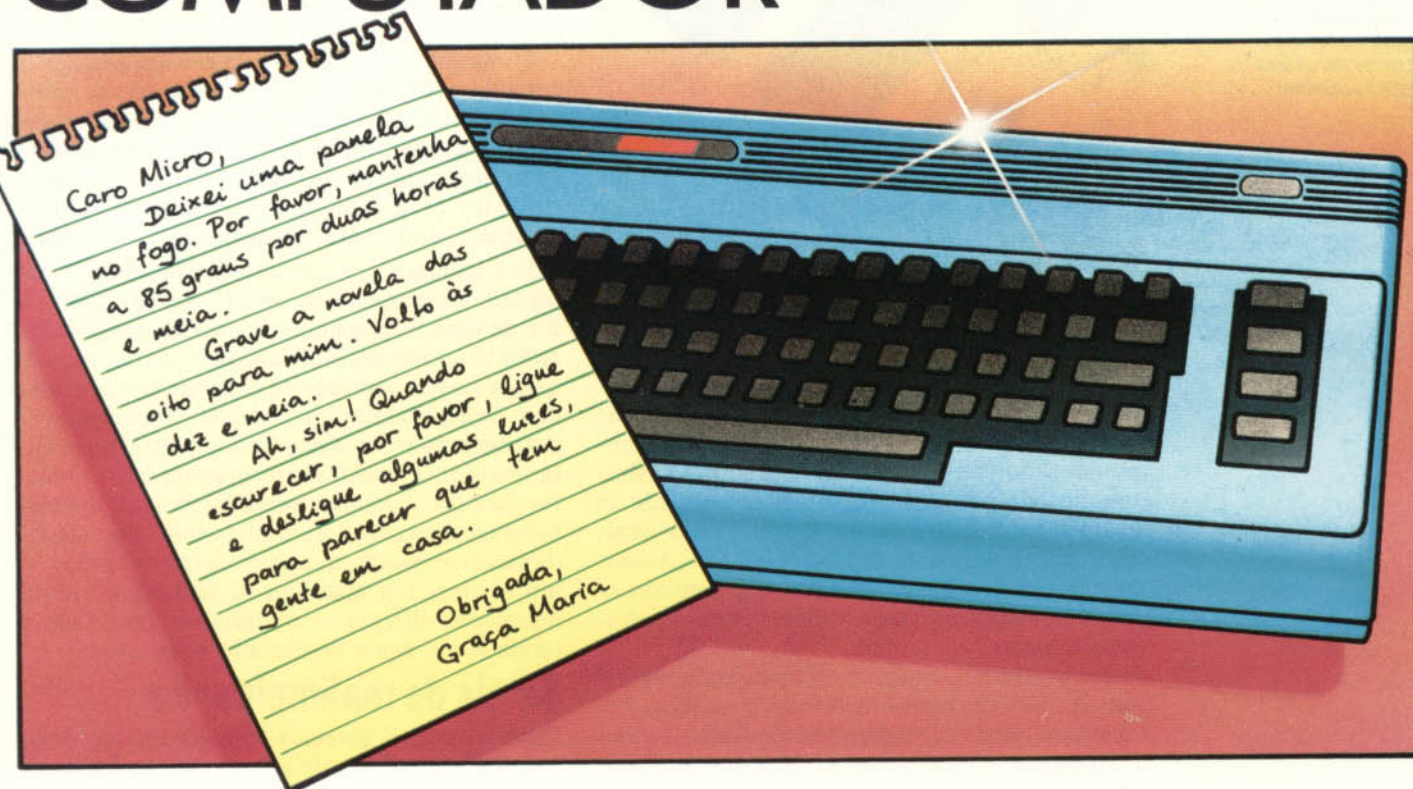
Embora deixasse de fabricar o TK 83 a partir de 1985, a empresa continuou lançando periféricos compatíveis com esse modelo e com o TK 85, entre os quais o gerador de som programável (PSG, Programmable Sound Generator) — capaz de criar inúmeros efeitos sonoros, de notas musicais a explosões —, além de interfaces para impressoras que utilizam o padrão Centronics.

Atendendo a profissionais liberais e empresas de médio porte, a Microdigital desenvolveu o TK 2000 Color, compatível com a linha Apple, com RAM de 64 Kbytes e conexão paralela para impressora, além de saída para unidade de disquete. A versão melhorada (TK 2000 II) foi apresentada ao público no ano seguinte, 1985. A Microdigital oferece também alguns periféricos para esse modelo: kit de conexão para impressora, joystick, interface para unidade de disquete de 5 1/4 polegadas e interface serial RS232C, que permite a comunicação via rede telefônica com bancos de dados, como , Cirandão e o Videotexto.

Em junho de 1985, a Microdigital introduziu no mercado o TK 90X, compatível com o ZX Spectrum, da Sinclair. Nesse modelo, que incorpora interface para joystick e caracteres de acentuação para a língua portuguesa, destacam-se os inúmeros recursos que o tornam apropriado para aplicações educacionais, inclusive permitindo a utilização da linguagem LOGO.

Outro lançamento de sucesso foi o modelo de videogame Onyx Jr., compatível com a tecnologia Atari e de design simples e funcional.

Os produtos da Microdigital devem parte de seu sucesso à grande quantidade de softwares disponíveis, muitos dos quais desenvolvidos pela Microsoft. Há programas aplicativos, utilitários, jogos inteligentes, jogos animados, programas de engenharia e educacionais. Com o lançamento do TK 90X (um micro apropriado para implementação em escolas), a Microdigital evidencia sua intenção de abranger também o público estudantil. A empresa pretende apresentar os equipamentos ideais para quem se inicia na informática e implantar esse recurso de aprendizagem nos estabelecimentos de ensino.

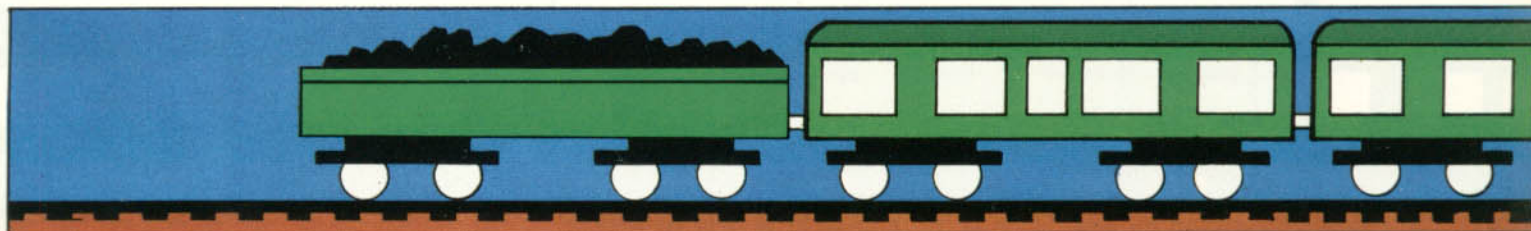
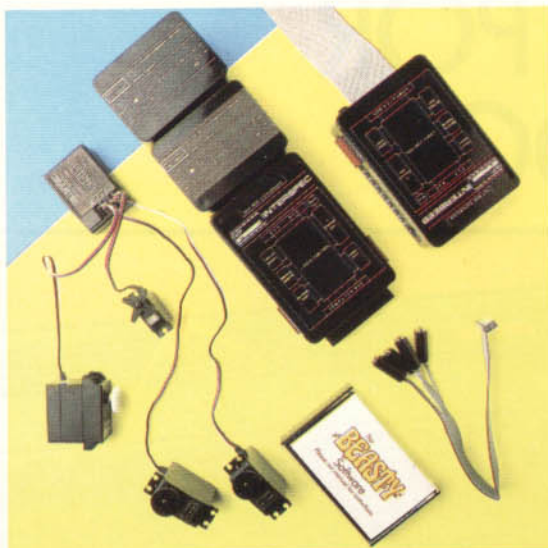
CONTROLE POR COMPUTADOR

Os usuários de micros têm a opção de desenvolver seu próprio software, mas poucos pensariam em construir periféricos em casa. Agora já existem componentes que tornam isso possível em alguns casos.

O único motivo pelo qual as pessoas programam um computador para abrir as cortinas da casa pela manhã ou regar as plantas quando saem de férias é que isso as diverte. E nada há de errado em fazer uma coisa pela simples razão de ser divertido. Quem escreve seus próprios programas por hobby não gasta horas e horas nisso apenas por prazer?

No momento, construir periféricos por conta própria não passa de uma brincadeira, mas em breve essa atividade poderá se revelar bastante lucrativa. Muita gente acredita que, num futuro próximo, robôs e outros dispositivos controlados por computadores serão parte de nossas vidas, da mesma forma como os micros tornaram-se lugar-comum a partir de 1980.

Portanto, a habilidade desenvolvida hoje poderá ter inestimável valor no futuro. Afinal, grandes indústrias de computadores, como a Apple e a Atari, começaram em garagens de pessoas que estavam apenas "brincando" com componentes e aparelhos eletrônicos.

Alguns elementos são indispensáveis em qualquer sistema controlado por computador. Além deste e do aparelho a ser controlado, precisa-se também de algum meio físico, para transmissão das mensagens de controle do micro ao aparelho, e de um software que decida quais serão essas mensagens. Isso é apenas o começo: o computador deverá ter, ainda, um meio de medir os efeitos de seu controle, para proceder aos ajustes necessários. Sem esse feedback (mecanismo de realimentação), o micro será tão inútil quanto um motorista de olhos vendados.

Automação doméstica
A conexão do micro a aparelhos domésticos permite a realização automática de inúmeras operações. Previamente programado, o computador reage a fatos como, por exemplo, queda de temperatura ou presença de ladrões ligando o aquecedor ou acionando o alarma.

Conexões de controle
Vários micros já dispõem de interfaces apropriadas para o controle de aparelhos. Em geral são dispositivos de chaveamento baseados em relés. O computador liga ou desliga a corrente de determinado equipamento e depois confirma, por um sensor, se ele está ou não funcionando.

Todos os sistemas controlados por computador funcionam com base em sinais elétricos. Mas os sinais (digitais) utilizados no interior do computador são fracos demais para uso direto. Mesmo uma simples lâmpada utiliza quantidade de energia consideravelmente maior do que qualquer sinal interno de um computador. Assim, precisamos de algum dispositivo que converta as "baixas voltagens" do computador em "voltagens maiores".

Num primeiro estágio, o computador necessita de canal de comunicação com o exterior. Para isso, alguns micros dispõem de saída — serial ou paralela — especial para o controle de dispositivos externos. Controladas pelo microprocessador, essas saídas (ou portas) possuem uma área reservada de endereçamento na memória. A leitura ou gravação, a partir do exterior, de algum dado nessa parte da memória não afeta o funcionamento do micro.

Sozinha, a saída para os dispositivos externos pode ser usada para acender e apagar LEDs (diodos emissores de luz), mas, para outras finalidades, são necessários mais componentes. Às vezes, convém acoplar alguns componentes eletrônicos e mais uma pequena fonte de energia elétrica para tornar possível o controle de relés. Um relé não passa de uma chave que liga e desliga correntes elétricas relativamente altas, mas pode ser controlado por um pulso elétrico de baixa intensidade. Constitui, portanto, uma das melhores maneiras de converter os fracos pulsos elétricos do computador em correntes utilizáveis.

Um sistema que permita utilizar relés está entre os primeiros objetivos de qualquer pessoa com intenção de controlar dispositivos eletromecânicos. Por meio de relés, acionam-se diferentes aparelhos: desde motores elétricos, bombas hidráulicas e campainhas, até trenzinhos e carrinhos elétricos radiocontrolados. A maior parte dos relés usados por entusiastas dos inventos caseiros só funciona com aparelhos movidos a pilha, mas isso é suficiente para a maioria dos projetos.

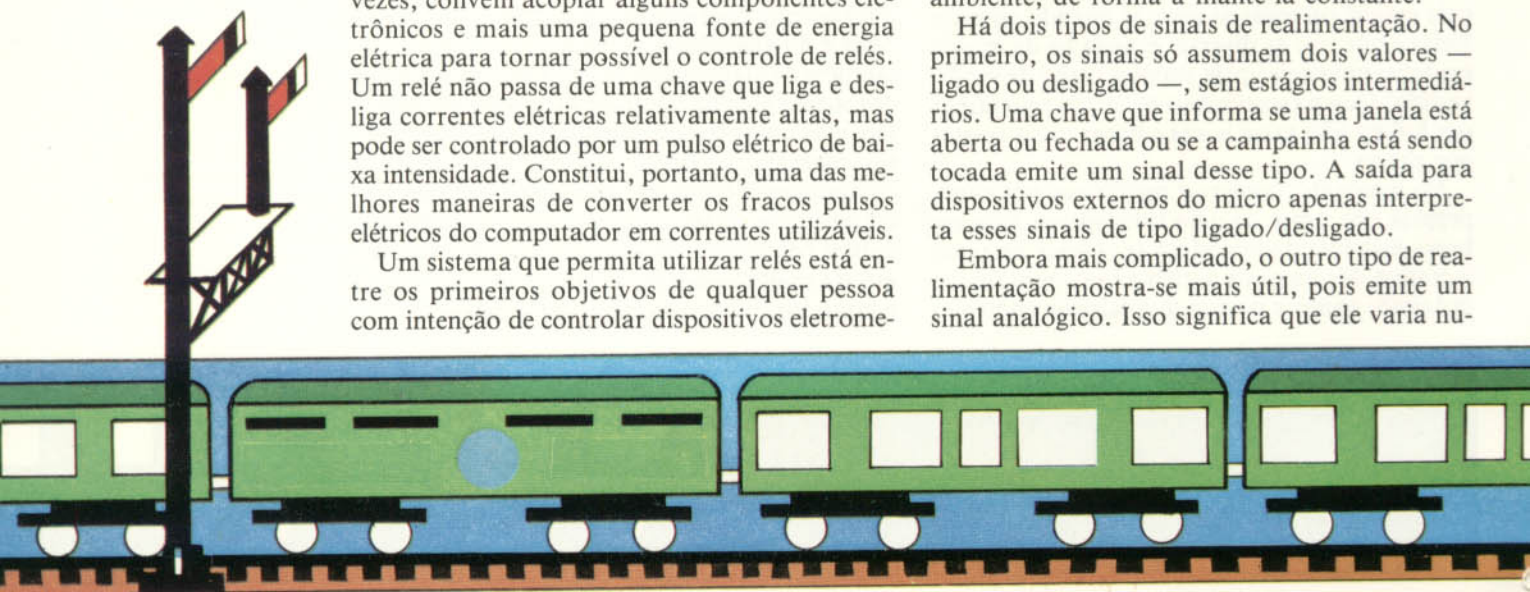
O chaveamento da corrente elétrica permite o controle, pelo computador, de aquecedores, lâmpadas potentes e inúmeros outros eletrodomésticos. Ele também possibilita ao computador atuar como timer, que liga a televisão no horário de um programa, ou acende e apaga as luzes da casa à noite para desencorajar os ladrões.

Por enquanto, o computador precisa ser conectado diretamente ao aparelho controlado. A fim de superar essa limitação, várias companhias estão desenvolvendo dispositivos para permitir que os fios condutores da corrente elétrica também transmitam as informações de controle. O sistema requer um computador central que emita sinais para unidades escravas em outros lugares da casa, ligadas a tomadas normais. Essas mensagens ligariam ou desligariam cada unidade escrava. Assim, qualquer eletrodoméstico — um abajur, um aparelho de televisão, ou um aquecedor elétrico — poderia ser conectado à unidade escrava e controlado pelo micro.

Sinais de realimentação

Quase todos os sistemas controlados por computador necessitam de algum tipo de feedback que avalie o funcionamento do conjunto. Aparentemente não há necessidade de realimentação quando o micro está apenas acendendo e apagando luzes. Mas seria muito melhor se ele pudesse detectar quando est. escurecendo e ligar as lâmpadas da casa em resposta a essa informação. Também seria muito útil se ele percebesse a entrada de alguém num aposento e acendesse as luzes. Ou, ainda, quando está controlando um aquecedor, ele precisa saber qual a temperatura ambiente, de forma a mantê-la constante.

Há dois tipos de sinais de realimentação. No primeiro, os sinais só assumem dois valores — ligado ou desligado —, sem estágios intermediários. Uma chave que informa se uma janela está aberta ou fechada ou se a campainha está sendo tocada emite um sinal desse tipo. A saída para dispositivos externos do micro apenas interpreta esses sinais de tipo ligado/desligado.

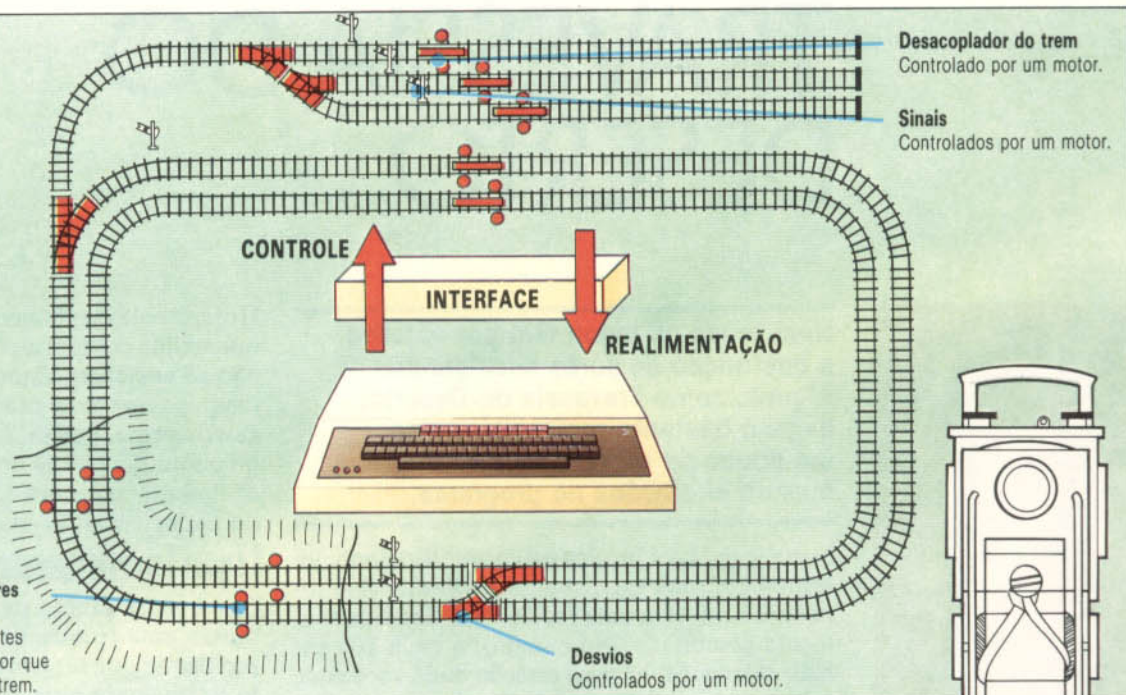
Embora mais complicado, o outro tipo de realimentação mostra-se mais útil, pois emite um sinal analógico. Isso significa que ele varia nu-

O chefe da estação

No controle de qualquer equipamento por micro — de um trenzinho elétrico a uma casa inteira — usa-se a mesma técnica. Monta-se um loop, no qual o computador emite sinais de controle, por meio de uma interface, ao dispositivo conectado. Quando esses sinais, que controlam servomotores, luzes etc. retornam, as informações são captadas por sensores como células fotoelétricas ou chaves liga/desliga. Essa realimentação permite que o micro controle o aparelho com precisão.

Desacoplador do trem
Controlado por um motor.

Sinais
Controlados por um motor.

LED/Pares de resistores fotossensíveis
Esses dois componentes constituem um detector que acusa a presença do trem.

Desvios
Controlados por um motor.

ma escala de valores, permitindo que seja usado para medir mudanças de temperatura, distâncias, rotação de objetos, o peso de alguma coisa ou a voltagem de uma bateria. Esse tipo de sinal é processado por um conversor analógico-digital (abreviado por A/D), que converte os valores variáveis do equipamento controlado (os sinais analógicos) em forma digital compreensível ao micro.

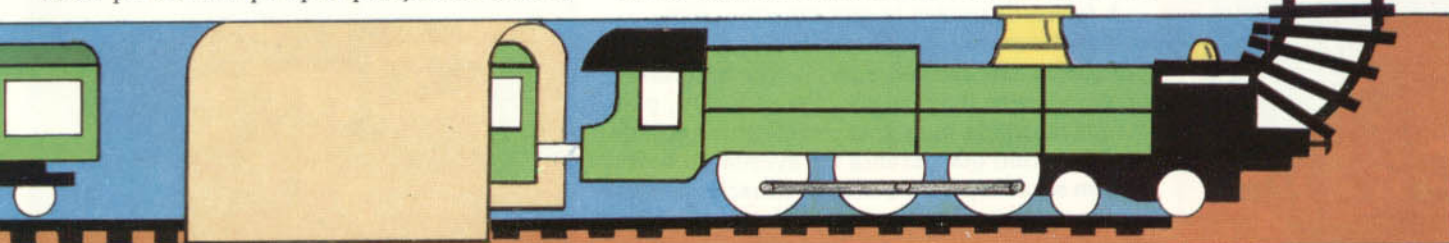
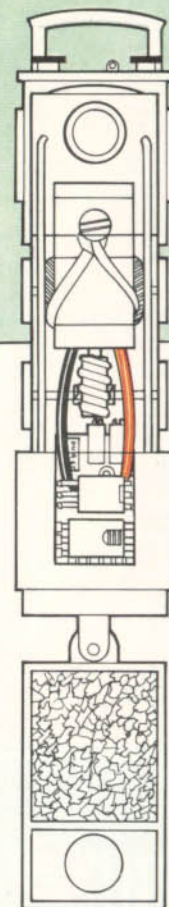
Resultado final

A existência de um dispositivo de realimentação faz grande diferença em tudo o que é controlado pelo computador. Se um motor está impulsionando uma roda, o micro pode calcular o quanto ela girou num dado tempo. Contudo, se existe algum peso sobre a roda, ou se as baterias que alimentam o motor estão quase descarregadas, um sensor óptico pode manter o computador informado dessas variações, notificando-o cada vez que a roda completa uma volta.

Motores elétricos desenvolvidos para serem controlados por computador têm uma espécie de realimentação embutida. O micro envia uma mensagem para que o motor se mova até determinada posição, na qual ele se desliga de modo automático. Há dois motores desse tipo: o gradual e o servomotor. O motor gradual funciona girando continuamente, de modo comum, e pode ser parado em qualquer posição. Já o servomotor só gira um pequeno ângulo — em geral, até 90 graus. Esse tipo de movimento é convertido em deslocamento vertical (puxa-empurra). Para que funcionem com computadores, ambos os motores exigem sistemas especiais de controle, e estes não estão disponíveis para muitos tipos de micro.

Na maioria das vezes, os servomotores são utilizados em braços mecânicos, tipo robô. Existem alguns braços robotizados pequenos que podem ser conectados a microcomputadores, mas são muito caros. É possível construir um braço robotizado com alguns servomotores a um custo mais baixo.

Outros tipos de dispositivos exigem voltagem variável para ser controlados por micros. Por exemplo, um pequeno motor elétrico que roda a diferentes velocidades, dependendo da voltagem a ele aplicada. O oposto de um conversor A/D é um conversor digital-analógico (D/A), que transforma os sinais digitais utilizados pelo computador em voltagens variáveis; por exemplo, para produzir sons, quando ligado a um alto-falante.

Usar um microcomputador para controlar outros equipamentos é como programar o próprio software. Você precisa combinar uma boa idéia, algum conhecimento técnico e muito tempo livre. Muitas vezes os resultados não compensam economicamente, mas o fazer-você-mesmo é muito mais divertido do que comprar pronto.

TRAVESSIA DO DESERTO

Nem todos os jogos têm por objetivo a destruição de seres alienígenas. Alguns, como Travessia do Deserto, exigem bastante raciocínio lógico, um pouco de experimentação e até mesmo alterações no programa.

Neste jogo, você precisa percorrer, dirigindo um caminhão, uma extensão de 1.000 km. Seu itinerário atravessa uma região desértica e inteiramente desabitada. No entanto, a cada 100 km mais ou menos, há uma estação onde você pode armazenar tonéis de combustível em seu veículo. Na base inicial, existe uma quantidade ilimitada de tonéis, mas cada um contém combustível suficiente apenas para o caminhão ir de uma estação até a outra.

A jornada não apresentaria problema se não fosse por um detalhe: o caminhão só pode transportar oito tonéis de cada vez. Desse modo, se você pretende chegar a seu destino, terá de apanhar combustível em vários pontos da rota, o que certamente implicará idas e vindas entre as diversas estações.

Em primeiro lugar, tome cuidado para não ficar sem combustível entre uma estação e outra — afinal, uma longa caminhada sob o sol escaldante do deserto não é perspectiva das mais atraentes. Ao contrário, pode até ser fatal.

O segundo objetivo do jogo é fazer com que toda a travessia seja realizada com o mínimo possível de combustível e percorrendo a menor quilometragem.

Você não encontrará muita dificuldade para solucionar o problema, pois o programa permite que o caminhão transporte oito tonéis de cada vez. Mas ele se tornará mais interessante se for alterado no sentido de exigir maior esforço mental.

O que aconteceria, por exemplo, se o caminhão carregasse apenas quatro ou seis tonéis de cada vez? Para investigar essas alternativas mude o valor da variável M na linha 60 e reconsidere o problema desde o início.

A forma de resolvê-lo será a mesma, como você logo descobrirá. Mas os intervalos entre os depósitos de combustível e as viagens necessárias de um a outro deverão ser alterados. Ou, para complicar ainda mais a questão, tente projetar um algoritmo que garanta a travessia do deserto com segurança, em qualquer situação. Um algoritmo desse tipo permite a elaboração de um programa que resolve seu problema específico. Hoje, a palavra "cibernética" designa a ciência que estuda comunicações e sistemas de controle não só nos organismos vivos como também em máquinas (os computadores, por exemplo). Originariamente, porém, *kybernetiké* indicava a arte do piloto. Travessia do Deserto permite que você "pilote" com auxílio do micro, transformando dados em ação otimizada.

Uma ferramenta de programação

Este quebra-cabeça de matemática e estratégia ilustra uma técnica valiosa para a solução dos problemas que surgem durante o desenvolvimento de qualquer programa. Primeiro, note como é importante experimentar com as informações disponíveis e testar algumas das possíveis variações. Se tudo correr bem, você acabará descobrindo um padrão geral, que inclui todas as ocorrências específicas. A partir dele, torna-se bem mais fácil projetar um algoritmo e, em seguida, desenvolver um programa abrangente.

Outra maneira de aperfeiçoar o jogo Travessia do Deserto é por meio do acréscimo de gráficos e outros recursos ou, ainda, estabelecendo dificuldades cada vez maiores — como a necessidade de transportar tonéis de água, além dos de combustível. No caso, inventar problemas pode ser tão estimulante quanto tentar resolvê-los.

Variações

O programa do jogo Travessia do Deserto, apresentado na página seguinte, foi escrito em Applesoft e, portanto, pode ser executado em todos os microcomputadores compatíveis com o Apple (inclusive o TK 2000). Se você quiser empregá-lo em algum equipamento da linha Sinclair (TK 85, CP 200 etc.), altere as linhas indicadas a seguir:

```
 110 CLS
 210 REM ( E L I M I N A R )
 220 REM ( E L I M I N A R )
 240 PRINT TAB (X-2);"▬";CHR$ (
128+CODE STR$ T);TAB (X-2);"▪ ▪"
 250 REM ( E L I M I N A R )
 320 PRINT "OH...OH... E▪ MELHOR
 VOCE       INICIAR NOVAMENTE."
 810 PRINT "VOCE E ";A;" TONEIS
FICARAM","NO DESERTO."
```

Para executá-lo num TK 90X, além das alterações acima, modifique a linha 240, escrevendo-a do seguinte modo:

```
 240 PRINT TAB (X-2);"▬";INVER
SE 1;T; INVERSE 0;TAB (X-2);"▪ ▪"
```

Programa básico

Linha Apple

Travessia do Deserto

```
10  REM  TRAVESSIA DO DESERTO
20  REM  INICIALIZACAO
30  DIM A(10)
40  LET A(1) = 80
50  LET S = 1
60  LET M = 8
70  LET N = 0
80  LET T = 0
90  REM  VIAGENS
110  HOME
120  PRINT "E)  1  2  3  4  5  6
      7  8  9'10"
130  PRINT "T) ";
140  FOR I = 1 TO 10
150  LET A$ =  STR$ (A(I))
160  IF  LEN (A$) ) = 2 THEN  GOTO
     190
170  LET A$ = " " + A$
180  GOTO 160
190  PRINT A$;" ";
200  NEXT I
210  PRINT
220  PRINT
230  LET X = 4 + (S - 1) * 3
240  PRINT "ESTACAO:";S
250  PRINT "TONEIS NO CAMINHAO:";
     T
260  IF S (  ) 10 THEN  GOTO 300
270  PRINT
280  PRINT "VOCE CONSEGUIU ..."
290  GOTO 740
300  IF  NOT (T = 0 AND A(S) = 0)
      THEN  GOTO 340
310  PRINT
320  PRINT "OH...OH... E MELHOR V
     OCE INICIAR": PRINT "NOVAMEN
     TE."
330  GOTO 740
340  PRINT
350  PRINT "SUAS OPCOES SAO:"
360  PRINT
370  IF T ) 1 THEN  PRINT "R RETI
     RAR TONEIS DO CAMINHAO"
380  IF A(S) ) 0 THEN  PRINT "P P
     EGAR TONEIS DE COMBUSTIVEL"
390  IF T ) 0 THEN  PRINT "A AVAN
     CAR UM ESTAGIO"
400  IF T ) 0 AND S ) 1 THEN  PRINT
     "V VOLTAR UM ESTAGIO"
410  PRINT
420  PRINT "QUAL E A ESCOLHA"
430  INPUT A$
440  IF T ) 1 AND A$ = "R" THEN  GOTO
     570
450  IF A(S) ) 0 AND A$ = "P" THEN
      GOTO 650
460  IF  NOT (T ) 0 AND A$ = "A")
      THEN  GOTO 510
470  LET S = S + 1
480  LET T = T - 1
490  LET N = N + 1
500  GOTO 110
510  IF  NOT (T ) 0 AND S ) 1 AND
     A$ = "V") THEN  GOTO 110
520  LET S = S - 1
530  LET T = T - 1
540  LET N = N + 1
550  GOTO 110
560  REM  DESCARREGAR TONEIS
570  PRINT
580  PRINT "QUANTOS";
590  INPUT A
600  IF (A ) T) OR (A (  )  INT (
     A)) OR (A ( 0) THEN  GOTO 59
     0
610  LET A(S) = A(S) + A
620  LET T = T - A
630  GOTO 110
640  REM   CARREGAR TONEIS
650  PRINT
660  PRINT "QUANTOS";
670  INPUT A
680  IF A ) A(S) OR A (  )  INT (
     A) OR A ( 0 THEN  GOTO 670
690  IF A + T ) M THEN  GOTO 670
700  LET T = T + A
710  LET A(S) = A(S) - A
720  GOTO 110
730  REM   FIM DO JOGO
740  PRINT
750  PRINT N;" TONEIS FORAM GASTO
     S."
760  PRINT T;" FICARAM NO CAMINHA
     O."
770  LET A = 0
780  FOR I = 2 TO 9
790  LET A = A + A(I)
800  NEXT I
810  PRINT "VOCE E ";A;" TONEIS F
     ICARAM NO DESERTO."
820  PRINT
830  PRINT "COMANDE ""RUN"" PARA
     TENTAR ","NOVAMENTE."
```

PRINTER/ PLOTTERS

O preço desses periféricos ultrapassa o orçamento da maioria dos usuários de micros, mas existem, no mercado internacional, modelos acessíveis e adequados ao uso pessoal.

Equipamentos complexos, os plotters são empregados para produzir gráficos detalhados em cores. Funcionam ainda como impressora, produzindo textos, e por isso são chamados também de "printer/plotters".

Dentro da unidade, há um tambor giratório com quatro pequenas canetas — esferográficas ou hidrográficas — de cores vermelha, azul, verde e preta. Para desenhar uma linha, o tambor gira, selecionando a cor desejada. Em contato com o papel, a caneta traça linhas horizontais (movimentos laterais do tambor) e verticais (movimentos do papel para cima e para baixo).

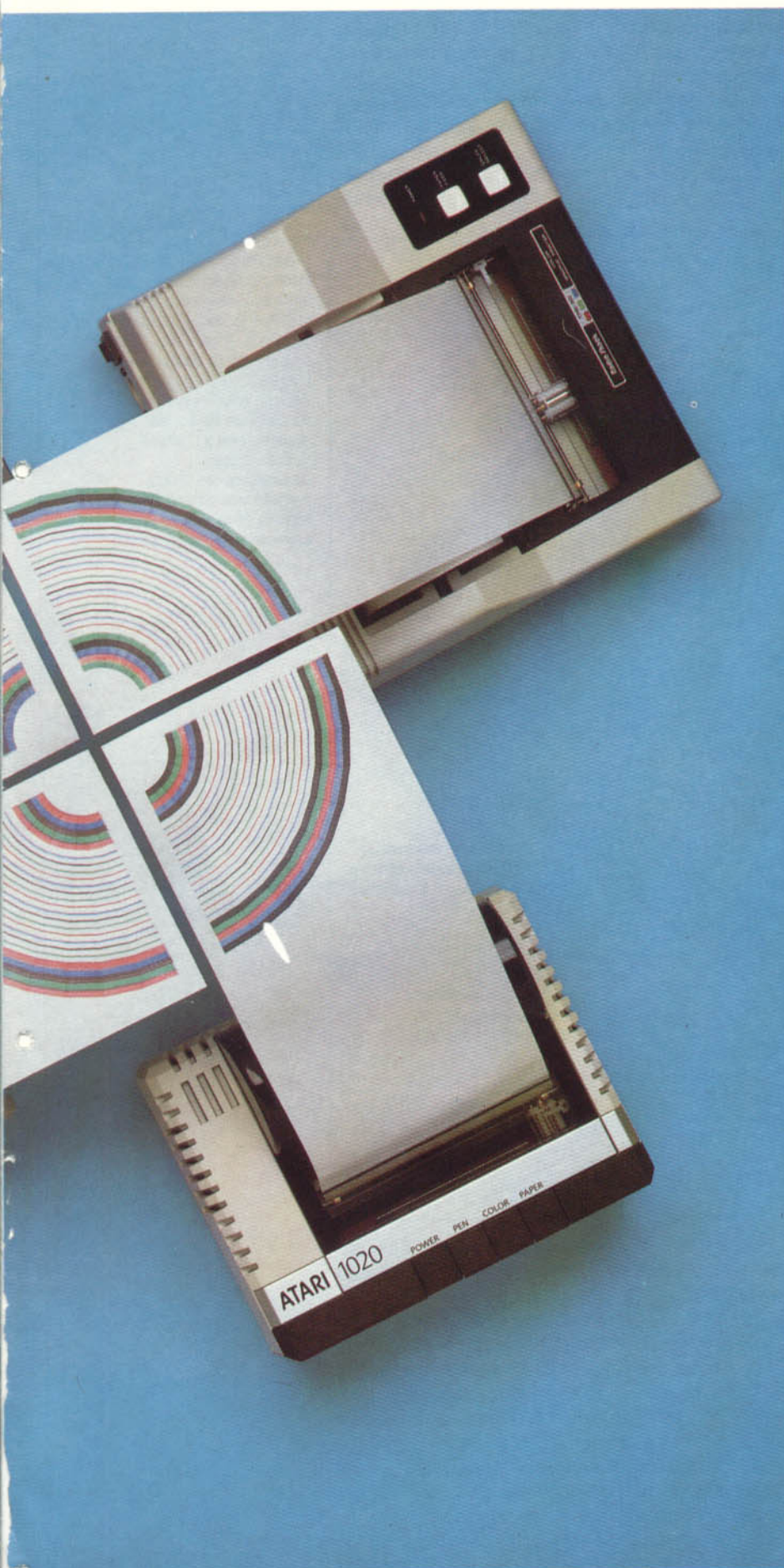
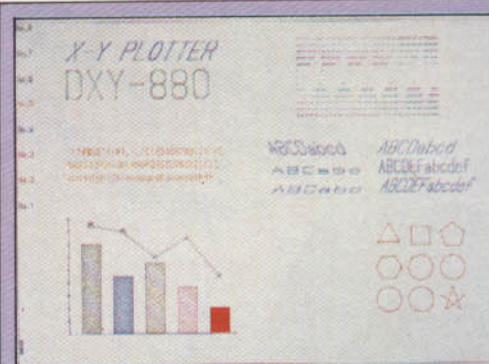
Os caracteres de texto são produzidos quase da mesma forma que os gráficos. O printer/plotter armazena os padrões das letras e outros caracteres em sua própria memória. Quando recebe um sinal do computador, a unidade apenas encontra o caractere requisitado na memória e o desenha como se fosse um padrão. A qualidade de impressão é muito boa, melhor que a da maioria das impressoras matriciais.

Em modo texto, o printer/plotter funciona da mesma forma que qualquer outra impressora. Embora o papel empregado por alguns modelos tenha apenas 11,5 cm de largura, a unidade produz quarenta ou oitenta caracteres por linha. Com papel estreito, um texto com oitenta caracteres resulta em letras pequenas, mas bastante nítidas.

É possível imprimir textos em qualquer das quatro cores, a dez caracteres por segundo. Essa velocidade mostra-se baixa, em relação às impressoras matriciais, mas aproxima-se à velocidade das impressoras tipo margarida.

Para produzir gráficos, o printer/plotter é chaveado para o modo gráfico, no qual produz linhas, curvas e letras grandes. Caracteres enviados para a unidade nesse modo não são impressos, pois no caso ela os interpreta como comandos. Quando se pretende traçar uma linha, precisa-se de comando, que em BASIC seria:

```
LPRINT "D 0,100,100,100,100,0,0,0"
```

A letra D é a instrução para traçar uma linha, e os números que se seguem dão a posição dos pontos pelos quais a linha vai passar. No caso,

Opção profissional
Nos plotters convencionais são utilizadas folhas de papel avulsas, adequadas a cada aplicação.

Recursos variados
Além de diagramas e gráficos, os plotters também podem funcionar como impressoras, "desenhando" os caracteres.

O printer/plotter DXY 880, com o qual foi produzido o desenho *acima*, é da fábrica japonesa Roland. Utiliza oito canetas e papel de 44 x 29 cm. Seu chip de ROM incorpora várias rotinas de desenho, que podem ser chamadas pelo programa que está rodando no micro: DRAW, CIRCLE, MOVE, HOME, LINE, SCALE, HATCHING etc. *À esquerda,* trabalhos com outros printers/plotters do mercado internacional.

supondo a cabeça de impressão localizada no início das coordenadas (0,0), o plotter desenhará um quadrado.

Considera-se a largura do papel dividida em 480 "espaços" horizontais, usados para posicionar o tambor. O papel utilizado pela unidade forma rolos de vários metros de comprimento. Isso pode dar a falsa impressão de não haver limite para a extensão vertical do desenho. Na verdade, porém, o deslizamento do papel pode resultar em linhas que não se unem conforme o desejado. Por isso, o printer/plotter impede que o papel seja puxado de volta além de certa distância.

É possível produzir gráficos complexos no printer/plotter, mas a programação necessária demora muito. Não há forma simples de copiar uma imagem diretamente da tela do computador, como fazem alguns aplicativos para impressoras matriciais. Desenhar uma linha é muito simples, mas a ausência de comandos como PAINT e FILL significa que, para preencher uma forma com cores, precisamos traçar muitas linhas contíguas, processo que demora e consome muita tinta.

Também não se pode empregar toda a largura do papel — uma margem deve ser deixada de cada lado.

O equipamento é mais adequado à produção de gráficos, dispondo de um comando que desenha automaticamente os eixos, já com as unidades requeridas marcadas nos intervalos apropriados.

Em modo texto, a unidade produz dois tamanhos de letras. Contudo, utilizada em modo gráfico, obtêm-se tamanhos de caracteres muito maiores, sendo possível imprimir no sentido lateral. Portanto, imprimem-se textos longos no sentido do comprimento do papel.

As canetas ficam sem tinta rapidamente, e podem ressecar quando deixadas na cabeça de impressão. Cada vez que se liga o equipamento, ele desenha um quadrado de cada cor, num teste automático das canetas.

Botões de controle
Semelhantes aos de controle do software.

Papel em rolo

Controlador das canetas
Empurra a caneta contra o papel durante a impressão.

Tambor giratório
Contém quatro canetas, esferográficas ou hidrográficas. Muda a cor de impressão por meio de movimentos rotativos.

Alimentação do papel
Um motor passo a passo, muito preciso, move o papel verticalmente em espaços de 0,2 mm.

Mecanismo de movimento horizontal
Desloca o tambor das canetas em espaços de 0,2 mm.

Papel e tambor
O printer/plotter tem uma resolução de aproximadamente 500 x 2.000 pixels no modo gráfico e vários tamanhos de caracteres. A cor é escolhida pela rotação do tambor das canetas. Obtém-se o texto ou o desenho pelos movimentos horizontal do tambor (da esquerda para a direita) e vertical do papel.
Os desenhos são de alta precisão, mas o movimento repetido do papel e do tambor provoca leve perda de registro.

BBC MODELO B

Boa qualidade técnica e grande disponibilidade de software, apoiadas no eficiente marketing da BBC, tornaram esse microcomputador muito popular na Inglaterra.

O BBC Modelo B é fabricado pela Acorn Computers, em Cambridge, e comercializado pela BBC (British Broadcasting Corporation, a rádio e tevê estatal britânica). Havia dois tipos, mas o primeiro deles, o Modelo A, mais barato e menos sofisticado, foi retirado de linha.

O Modelo B teve boa aceitação nas escolas e conta com o aval do governo como material pedagógico. Considerável software educacional foi desenvolvido para esse equipamento — desde linguagens de programação até pacotes para aprendizagem apoiada em computador.

Os recursos técnicos do BBC Modelo B ainda são considerados excelentes, apesar da grande variedade de novas máquinas lançadas mais recentemente no mercado inglês. A linguagem de programação BBC BASIC, em particular, dispõe de amplo leque de comandos para operar com funções especiais, o que facilita a tarefa de desenvolver e editar programas.

O equipamento possibilita oito modos gráficos distintos. Isso significa que o usuário pode escolher a resolução entre baixa, média e alta, apesar de, no último caso, o número de cores disponíveis ficar limitado. O máximo de resolução que se consegue é de 640 x 256. Muitos usuários optam por ligar a CPU a um televisor comum, mas recomenda-se um monitor de vídeo dedicado, a fim de se obterem os melhores resultados dos gráficos do Modelo B. Há comandos para desenhar linhas, círculos e construir uma série de imagens na tela.

Seu design, em função da fonte de alimentação interna, apresenta-se bem-acabado e compacto, mas as interfaces — na parte posterior e na inferior do gabinete — são mais numerosas do que em outros equipamentos. Isso significa que grande variedade de dispositivos para expansão pode ser encontrada, inclusive várias marcas de unidades de disco não fabricadas especialmente para o BBC.

Além das interfaces para unidades de disco, impressora e um dispositivo analógico para controle de equipamentos de laboratório ou instrumentos de medida, o Modelo B pode facilmente trabalhar em rede. Isso o torna ideal para salas de aula, onde vários usuários podem partilhar a mesma unidade de disco ou impressora.

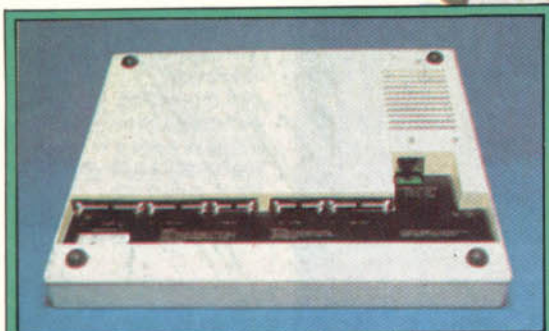
Teclado funcional
Um dos pontos fortes do BBC Modelo B. Com bom layout, dispõe de vários recursos, além de acabamento sólido. Pessoas acostumadas com teclado de máquinas de escrever comuns sentem-se à vontade.
As quatro teclas com setas, ao lado direito, destinam-se a movimentar o cursor na tela, para editar textos ou programas. As dez teclas vermelhas, de funções programáveis, são especialmente úteis para programas educacionais, pois o usuário tem apenas de escolher a resposta certa em dez alternativas.
Uma característica interessante é a inclusão de três LEDs para indicar se o motor do cassete está funcionando e se as teclas [Shift] ou a trava das maiúsculas foram acionadas.

Unidade de disco
A unidade de disco do BBC Modelo B é um acessório muito atraente, mas custa caro e armazena apenas 100 Kbytes. Existem outras marcas e a preços inferiores que devem ser levados em consideração.

Entrada analógica
Possibilita ao computador ler sinais analógicos — por exemplo, de um sensor de calor. Mais usada em laboratórios ou trabalhos experimentais.

Controlador de vídeo
Este chip obtém dados provenientes da RAM e os converte em sinais de vídeo.

Chips estrangeiros
Observando cuidadosamente estes chips, você perceberá que a fabricação de microprocessadores é uma questão internacional.
O BBC Modelo B contém chips fabricados em Malásia, Japão, Portugal, Escócia e Estados Unidos.

Processador de interfaces
Adaptadores versáteis de interface como este 6522, de tecnologia MOS, cuidam da ligação com dispositivos externos.

Saídas para periféricos
Parte inferior do BBC Modelo B, vendo-se as tomadas onde se conectam os periféricos.

Porta para impressora
Liga qualquer impressora paralela.

Porta digital
Para testes com dispositivos digitais e circuitos lógicos feitos pelo próprio usuário.

Interface para cassete
Programas em gravador cassete comum são guardados de duas formas: em velocidade máxima ou com maior fidelidade da gravação.

Saída RGB
Fornece sinais separados para os componentes vermelho, verde e azul do sinal do vídeo. Controla um monitor em cores de alta qualidade.

Saída para televisor
Para ligar à antena do aparelho.

Saída para vídeo composto
Para utilização com um monitor monocromático ou colorido.

Entrada RS232C
Interface serial de alta velocidade para utilização com vários periféricos.

Modulador
Conduz o sinal de cores do controlador de vídeo e o converte em saída adequada para o televisor.

Cristal de quartzo
Produz os pulsos de clock, sincronizando todas as operações.

Microprocessador
O 6502 de tecnologia MOS executa todo o trabalho de processamento.

Memória para o usuário
O BBC Modelo B contém 32 Kbytes de RAM para armazenamento de programas, dados e apresentações gráficas.

ULA (Uncommitted Logic Array)
Matriz de lógica não determinada. Executa o trabalho que em outros computadores exigiria dezenas de componentes. A peça de metal na parte superior age como isolante térmico e protege o chip contra superaquecimento e radiofreqüência.

ROM
Estes dois chips de ROM fornecem a linguagem de programação BASIC e o sistema operacional, que é o conjunto de programas necessários para que o computador controle todas as funções internas.

Tube
Interface projetada especialmente para que o BBC opere com microprocessadores alternativos.

BBC Modelo B

DIMENSÕES
409 x 358 x 78 mm.

PESO
3,7 kg.

MICROPROCESSADOR
6502A.

CLOCK
2 MHz.

MEMÓRIA
32 Kbytes de RAM; 32 Kbytes de ROM, incluindo BASIC e sofisticado sistema operacional.

VÍDEO
Oito modos gráficos, com ampla escolha de displays. Área máxima de texto: 32 linhas de 80 caracteres. Resolução gráfica de 640 x 256 pixels. Em baixa resolução, mais de dezesseis cores.

TECLADO
Tipo máquina de escrever, com 74 teclas, incluindo dez de funções programáveis e quatro para controle do cursor.

LINGUAGENS
BASIC, LISP, FORTH, LOGO.

INTERFACES
Televisor, monitores monocromáticos e em cores, unidade de disco, impressora, entrada analógica, porta digital e Tube.

DOCUMENTAÇÃO
O guia do usuário parece presumir que seus leitores já têm familiaridade com computadores. Extensos capítulos são dedicados à utilização especializada de programas que controlam gráficos sofisticados, som e dispositivos de entrada/saída. Está incluída uma explicação detalhada e abrangente do funcionamento e programação do microprocessador 6502.

DIMENSÕES MATRICIAIS

Graças à flexibilidade de suas estruturas de dados, o PASCAL não depende tanto das matrizes quanto outras linguagens. Neste artigo, veremos seus recursos de compactação e dimensionamento de matrizes.

Em muitas linguagens de programação, considera-se a matriz o método mais prático para representar dados do mundo real — por exemplo, listas e tabelas — numa estrutura de dados compreensível pelo computador. Outra vantagem é que ela permite acessar, para processamento, elementos isolados. No entanto, deve-se principalmente ao hábito seu emprego quase universal. Além das matrizes, não havia outro método de estruturação de dados quando se desenvolveram as primeiras linguagens de alto nível. O FORTRAN, por exemplo, possuía somente números complexos e matrizes; e seu descendente, o BASIC, omitia os números complexos. Por esse motivo, o loop FOR-NEXT é a única estrutura definida de iteração.

Isso não constituía problema quando o FORTRAN era empregado apenas para executar operações com matrizes em cada elemento de uma tabela, ou quando empregava-se o BASIC para familiarizar os estudantes com a rigorosa programação do FORTRAN. Já vimos alguns exemplos da extraordinária flexibilidade de controle que os loops condicionais (WHILE e REPEAT) oferecem ao PASCAL. E — talvez ainda mais importante — começamos agora a compreender o enorme poder e a facilidade de expressão que oferecem algumas das outras estruturas de dados do PASCAL, como conjuntos e registros. Desse modo, o predomínio das matrizes como uma "ferramenta de dados" universal para programação é consideravelmente reduzido no PASCAL.

Versatilidade das matrizes

Embora a maioria dos programadores sinta-se em terreno familiar ao lidar com matrizes no PASCAL, há dois pontos importantes que devem ser ressaltados. Como essa linguagem utiliza vários métodos de estruturação de dados, muitos dos problemas de programação são resolvidos de maneira mais eficiente por meio de outras estruturas. Em segundo lugar, as matrizes não requerem qualquer limitação quanto à complexidade estrutural de seus elementos, de modo que se pode obter uma flexibilidade muito maior.

A definição de um tipo de matriz exige a reserva de memória para um tamanho determinado de matriz e define tanto o tipo base de seu índice (indexador), como o tipo de cada componente. Assim, por exemplo,

```
TYPE
  CONTACARATER = ARRAY ['A' .. 'Z'] OF
    INTEGER;
VAR
  LISTA : CONTACARATER;
```

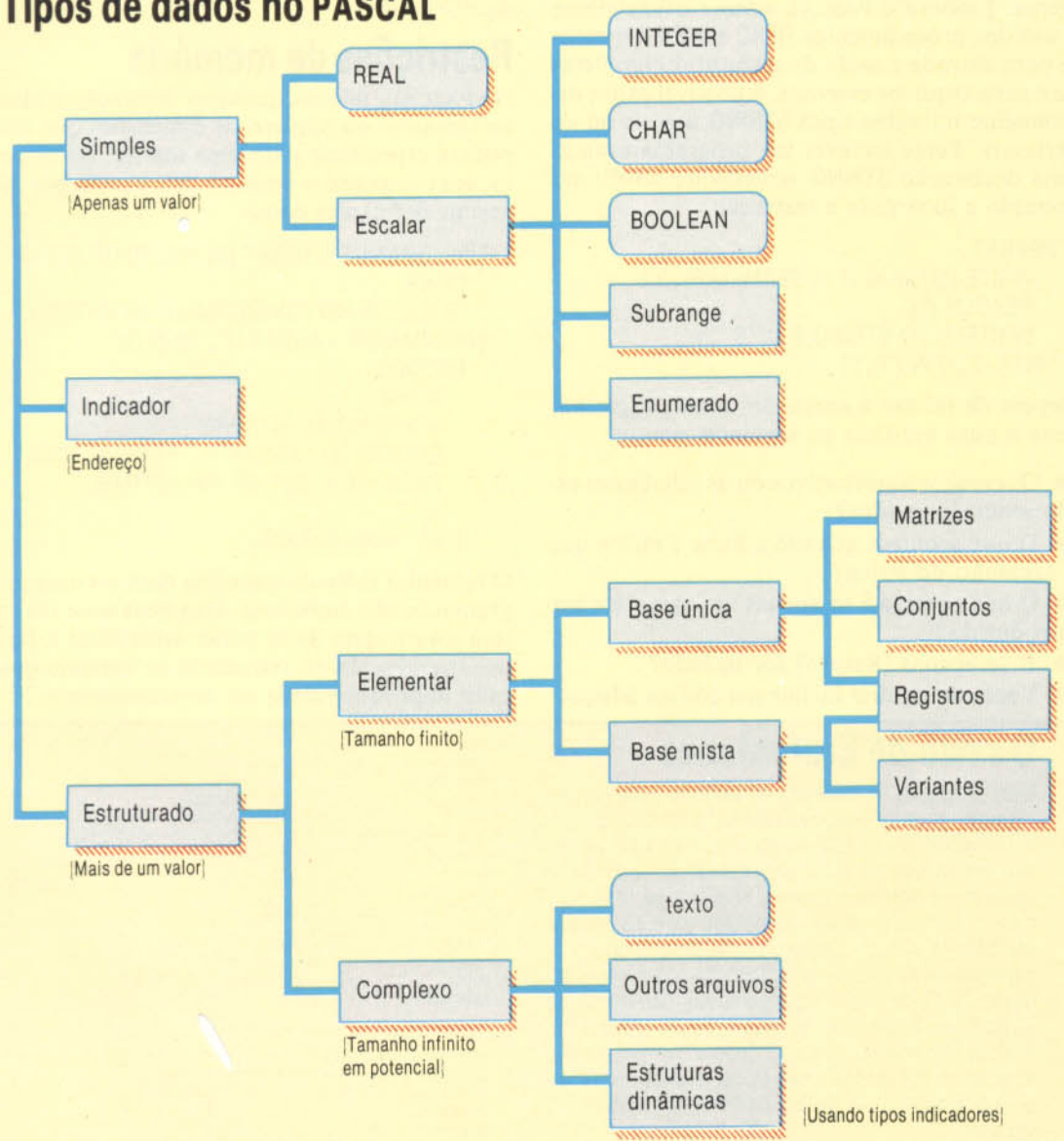
reserva espaço de armazenamento para 26 valores integer, indicados pela especificação do identificador da matriz, seguido por uma expressão de indexação entre colchetes. Os colchetes são sempre usados com matrizes, como no BASIC original (o uso subseqüente dos parênteses foi adotado em função de alguns conjuntos de caracteres incompletos, que não possuíam colchetes). Para acessar um determinado valor integer na lista do exemplo, escrevemos

```
LISTA ['M'] ou LISTA [PRED (SIMBOLO)]
```

No segundo exemplo, a expressão PRED (SIMBOLO) deve resultar num valor CHAR na faixa de 'A' a 'Z'. Emprega-se qualquer tipo escalar verdadeiro para indexar dimensões matriciais, mas não tipos reais ou estruturados. Isso garante a inexistência de referências absurdas como, por exemplo, LISTA ['SEGUNDO'] ou FLAG [3,75]. Com a definição de tipo exemplificada acima, inicializamos cada elemento de CONTADOR em zero numa matriz de tipo CONTACARATER:

```
VAR
  LETRA : CHAR;
  CONTADOR : CONTACARATER;
BEGIN
  FOR LETRA := 'A' TO 'Z' DO
    CONTADOR [LETRA] := 0
```

e assim por diante. Seriam ilegítimos, para essa estrutura, tanto CONTADOR [1], pois o tipo de índice está errado, como CONTADOR [A], que não existe. Assim, sempre será um bom procedimento definir o tipo do identificador para as variáveis de indexação. De qualquer forma, são invariavelmente necessários para definição de procedimentos e funções.

Se essas faixas estiverem definidas por valores introduzidos em uma definição CONST, será possível reformular todo o programa — modificando-se apenas uma linha — para que ele opere com estruturas de tamanhos diferentes. Por exemplo, embora não haja o tipo STRING pré-declarado no PASCAL, um modo de criá-lo (embora não necessariamente o melhor) seria

Típico demais!
A rigorosa estruturação de dados no PASCAL obriga a uma abordagem mais metódica na elaboração de um programa. Apenas dois, dentre os três tipos de dados mostrados no esquema, subdividem-se em outras categorias. Mas a variedade possível de dados simples e estruturados exige definições estritas de todos os dados do programa. Esse esforço adicional resulta em soluções mais elegantes para os problemas de programação do que as obtidas em outras linguagens.

```
CONST
  MAXTAMANHO = 25;
TYPE
  TAMSTRING = 1 .. MAXTAMANHO;
  STRING = PACKED ARRAY [TAMSTRING] OF
    CHAR;
```

Note que a palavra reservada PACKED precede qualquer palavra reservada de tipo estruturado (SET, ARRAY, RECORD ou FILE). "Compactando" as estruturas de dados, obtemos vantagens consideráveis no uso da memória, especialmente em computadores com palavras longas.

O tipo string

A única ocasião em que você realmente precisa se preocupar com a compactação é ao usar constantes de strings literais. Um string de caracteres entre aspas é indexado a partir de 1 (e não de 0); portanto, declara-se implicitamente como:

```
PACKED ARRAY [1 .. N] OF CHAR
```

onde N é o número de caracteres no string. Considere este programa:

```
PROGRAM PACK STRING;
CONST
  PASCAL = 'PASCAL';
TYPE
  STRING = PACKED ARRAY [1 .. 10] OF CHAR;
VAR
  S      : STRING;
BEGIN
  S := PASCAL           ;
  S := 'EXCESSIVOS CARACTERES';
  S := 'PASCAL         '
END.
```

No núcleo do programa, as duas primeiras instruções são ilegais devido à incompatibilidade dos tamanhos, mas a terceira é admissível (quatro espaços usados como caracteres de preenchimento). Se a definição do tipo STRING não estivesse compactada, todas as atribuições para literais seriam incompatíveis. Na prática, contu-

do, essas restrições não constituem grande problema. Embora o PASCAL admita oficialmente o uso dos procedimentos READ e WRITE somente para entrada e saída de estruturas completas de e para arquivos externos, é possível exibir diretamente no vídeo tipos STRING literais ou de variáveis. Tente escrever um programa usando uma declaração STRING semelhante ao último exemplo e incorpore a instrução

```
REPEAT
  WRITE ('STRING (T P/ TERMINAR) : ');
  READLN (S);
  WRITELN ('O STRING E : "'' S,''")
UNTIL S [1] IN ['T','t']
```

Depois de iniciar a execução desse programa, teste-o para verificar os seguintes pontos:

- Os espaços significativos ou as tabulações estão sendo ignorados?
- O que acontece quando a linha é maior que o tamanho do string?
- O que conterá S se apenas um caractere for introduzido?
- E se apenas [Return] for teclado?
- Você conseguiria incluir um código adequado para preencher com espaços elementos não significativos?

Restrições de memória

No PASCAL, não há qualquer limitação quanto ao tamanho ou número de dimensões que você poderá especificar para uma matriz. No entanto, seu computador pode ser desculpado por não aceitar definições como

```
TIPO GRANDE = ARRAY [INTEGER] OF SET OF
  CHAR;
  |SOLICITA NO MINIMO 64K x 128 BYTES!|
AINDAMAIOR = ARRAY [1 .. 1000] OF
  RECORD
    NOME,
    SOBRENOME : STRING;
    ENDERECO : ARRAY [1 .. 5] OF STRING;
    PRESENCA : SET OF PALESTRAS;
    |ETC.|
  END; |AINDAMAIOR|
```

O tamanho físico da memória limitará uma programação tão ambiciosa. Isso mais uma vez reitera a vantagem de se poder especificar a faixa dos tipos escalares, reduzindo ao máximo qualquer improdutividade no armazenamento.

O crivo de Eratóstenes

Este programa — baseado no método para gerar números primos, desenvolvido pelo filósofo e matemático grego Eratóstenes (séc. III a.C.) — tem sua velocidade de execução aumentada quando se ignoram os números pares e emprega-se uma matriz de "flags" (8.192 elementos para os primos até 16.384). Assim, torna-se necessária uma capacidade de memória de pelo menos 8 Kbytes; talvez 32 Kbytes nas linguagens que não possuem operadores booleanos de byte único e integers de 4 bytes. No PASCAL, contudo, todo o conjunto exige apenas 16.384 bits (2 Kbytes), permitindo expressar o algoritmo original de Eratóstenes com mais clareza:

```
COLOQUE TODOS OS NUMEROS (1 .. MAX) NUM CRIVO
TOME O NUMERO 1 (PRIMO POR DEFINICAO)
|IMPRIMA SE DESEJADO|
REPEAT
  PEGUE O MENOR NUMERO DO CRIVO
  |IMPRIMA ESSE PRIMO|
  REMOVA TODOS OS SEUS MULTIPLOS DO CRIVO
UNTIL (O CRIVO ESTEJA VAZIO)
```

Um conjunto assim só pode ser manipulado por computadores de grande porte, mas é possível uma matriz de conjuntos menores (com 100 ou 1.000 elementos, dependendo de seu compilador). Determina-se o índice de matriz de cada conjunto por meio de uma divisão integer, e representa-se sua inclusão no conjunto por N MODULO 100 ou 1.000.

```
PROGRAMA     PRIMOS     (OUTPUT);
|ENCONTRA NUMEROS PRIMOS PELO ALGORITMO
  DE ERATOSTENES|
CONST
  TAMSET       =      100;   |IMPLEMENTACAO|
  PREDTAMSET   =       99;   |TAMSET - 1|
  MAIORPRIMO   =    16383;   |PARA MAXIMTS = 32767|
  LISTAMAX     =      163;   |MAIORPRIMO DIV TAMSET|

TYPE
  FAIXAPRIM    =   1 .. MAIORPRIMO;
  DIMENSAO     =   0 .. LISTAMAX;
  FAIXASET     =   0 .. PREDTAMSET;
  CRIVO        =   SET OF FAIXASET;
  ERATOSTENES  =   ARRAY [DIMENSAO] OF CRIVO;
  CARDINAL     =   0 .. MAXINT;

VAR
  CRIVOS        :   ERATOSTENES;
  CONTADOR,
    MULTIPLO    :   CARDINAL;
  INDICE        :   DIMENSAO;
  N             :   FAIXAPRIM;
  NUMERO        :   0 .. PREDTAMSET;

BEGIN
  WRITELN;
  WRITELN ('CRIVO DE ERATOSTENES' : 50);
  WRITELN ('= = = = = = = = = = = = =' : 50);
  WRITELN;
  WRITELN;

  FOR INDICE := 0 TO LISTAMAX DO
    CRIVOS [INDICE] := [ 0 .. PREDTAMSET];
      |COLOCA TODOS OS NUMEROS NO CRIVO|

  CRIVOS [0] := [2 .. PREDTAMSET];
  |WRITELN (1);| |PRIMO POR DEFINICAO|
  CONTADOR := 1;
  FOR N := 2 TO MAIORPRIMO DO
    BEGIN
      INDICE := N DIV TAMSET;
      NUMERO := N MOD TAMSET;

      IF NUMERO IN CRIVOS [INDICE] THEN
        BEGIN
          CONTADOR := SUCC (CONTADOR);
          |WRITELN (N);|
          CRIVOS [INDICE] := CRIVOS
            [INDICE] - [NUMERO];
          MULTIPLO := N + N;

          WHILE MULTIPLO < = MAIORPRIMO DO
            BEGIN
              INDICE := MULTIPLO DIV TAMSET;
              CRIVOS [INDICE] := CRIVOS [INDICE]
                -[MULTIPLO MOD TAMSET];
              MULTIPLO := MULTIPLO + N
            END
        END
    END;

  WRITELN;
  WRITELN (CONTADOR : 25,'PRIMOS ENCONTRADOS.')

END
```

APLICATIVOS

Advogados, médicos, proprietários de restaurantes e outros profissionais necessitam de programas específicos — os aplicativos —, que são adaptados de pacotes de software genérico ou, então, desenvolvidos especialmente.

Apesar da quantidade de programas disponíveis no mercado, a maioria apresenta um grande inconveniente: é abrangente demais. Assim, existe pouca probabilidade de que sejam imediatamente aplicáveis a um problema específico. Duas alternativas apresentam-se ao usuário: adaptar o problema ao software existente — o que pode não ser satisfatório —, ou modificar o programa de modo que atenda a suas necessidades.

Uma terceira possibilidade, cada vez mais difundida na Europa e nos Estados Unidos, são as aplicações verticais ou softwares aplicativos verticais, isto é, a criação, por entusiastas da informática ou por firmas especializadas, de programas para solução dos problemas específicos de certos profissionais.

Em São Paulo, especialistas em desenhos por computador desenvolveram um programa que ajuda dentistas a planejar tratamentos ortodônticos. A partir da descrição da arcada dentária do paciente, o programa desenha a involução provável dos defeitos sob condições de uso de aparelhos corretores.

Outro exemplo de software vertical é o aplicativo desenvolvido pelo arquiteto José Eduardo Maluf de Carvalho. Para realizar o projeto de um edifício, o arquiteto ou o engenheiro precisa saber que características a obra deverá ter em obediência às leis de zoneamento.

Cada uma das dezoito zonas em que se divide a cidade de São Paulo possui uma legislação precisa quanto ao tipo de construção permitida, áreas máxima e mínima de construção, recuos etc. Em geral são necessárias três horas para se realizar esse levantamento; mas, com auxílio do programa criado por José Eduardo, abrevia-se esse tempo para apenas cinco minutos.

Essas são algumas respostas criativas à inevitável questão colocada por pessoas que acabaram de adquirir um micro: "Para que ele serve?" Estima-se que, em média, os usuários de micros não utilizam mais do que 10% da capacidade do equipamento. Afinal de contas, investir dinheiro numa máquina e limitá-la a uma única aplicação — seja jogos ou administração de contas pessoais — não é, certamente, tão razoável e divertido quanto explorar ao máximo sua versatilidade.

Perfis do mercado
Os aplicativos respondem às necessidades de grupos específicos de usuários e são vendidos a grandes setores do mercado. No diagrama, essas áreas de aplicação relacionam-se aos interesses de uma parcela significativa dos usuários. O eixo vertical representa o nível de complexidade da aplicação — os mais baixos são os métodos manuais, a seguir vem o software pronto e, por fim, o software aplicativo. Os perfis de uso mostram que o software pronto se presta muito bem aos requisitos comerciais, mas também indicam que usuários especializados precisam adaptar os pacotes de software já existentes ou criar os seus próprios.

Visão vertical

Hierarquia das aplicações
1 SISTEMAS MANUAIS
2 SISTEMAS MANUAIS/MECÂNICOS
3 PACOTES COMERCIAIS
4 SOFTWARE DE USO GERAL ADAPTADO
5 PACOTES COMERCIAIS DE USO ESPECÍFICO
6 PACOTES DE USO ESPECÍFICO DESENVOLVIDOS PARA OU POR UM USUÁRIO

EDUCATIVO
FINANCEIRO
DIVERTIMENTO
PESSOAL
MÉDICO
LEGAL
PROJETO/DESENVOLVIMENTO
VAREJO
PROFISSIONAL
GERÊNCIA
VENDAS
OPERACIONAL
FINANCEIRO
ADMINISTRATIVO
CAD
PROCESSADOR DE TEXTO
BANCO DE DADOS
PLANEJAMENTO/CONTABILIDADE

NO NÍVEL

Este curso, que já permite projetar circuitos complexos, mostra agora o desenvolvimento de todo o processo de desenho para um circuito gerador de bits de paridade e um decodificador de prioridades.

Antes de examinar o projeto desses dois aplicativos de nível adiantado, convém analisar em detalhe outra porta lógica importante — a porta OU exclusivo (XOU). Embora essa porta já seja conhecida, ainda não foram investigados nem a álgebra booleana, nem os símbolos do diagrama de circuito para ela:

Tabela de validação

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

Símbolo do circuito

A — XOU — C
B —

Símbolo booleano = ⊕

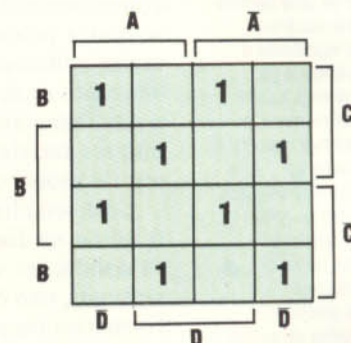
Pela tabela de validação, nota-se que a saída C pode ser expressa de duas formas:

a) $C = A \oplus B = \overline{A}.B + A.\overline{B}$

b) $\overline{C} = \overline{A \oplus B} = \overline{A}.\overline{B} + A.B$

Forma-se a segunda expressão considerando-se os casos em que C não é 1 (ou seja, é 0). Essa porta terá uso no primeiro aplicativo.

Gerador de bits de paridade

A paridade é um conceito importante no projeto de sistemas de transmissão de dados. O bit de paridade (ver o diagrama) de um código binário é acrescentado ao restante do código para fazer com que todos os códigos transmitidos tenham um número par (paridade par) ou ímpar (paridade ímpar) de dígitos 1. O bit de paridade funciona como um sistema de verificação para nos certificarmos de que houve transmissão correta. O circuito desenhado admite um código de 4 bits e apresentará o bit de paridade apropriado. Com uma pequena modificação, o circuito também poderá funcionar como verificador de paridade de dados recebidos. A tabela de validação para esse circuito encontra-se na margem. A representação desses valores em um mapa-k é

O padrão simétrico apresentado pelo mapa-k não permite simplificação porque não é possível formar grupos. A expressão resultante para P será

$$P = \overline{A}.\overline{B}.\overline{C}.D + \overline{A}.\overline{B}.C.\overline{D} + \overline{A}.B.\overline{C}.\overline{D} + \overline{A}.B.C.D + A.\overline{B}.\overline{C}.\overline{D} + A.\overline{B}.C.D + A.B.\overline{C}.D + A.B.C.\overline{D}$$

Agrupando os termos em vermelho e em azul, simplificamos a expressão para

$$P = (\overline{A}.\overline{B} + A.B).(\overline{C}.D + C.\overline{D}) + (A.\overline{B} + \overline{A}.B).(\overline{C}.\overline{D} + C.D)$$

Substituindo as expressões para a porta XOU apresentadas no início, obtemos

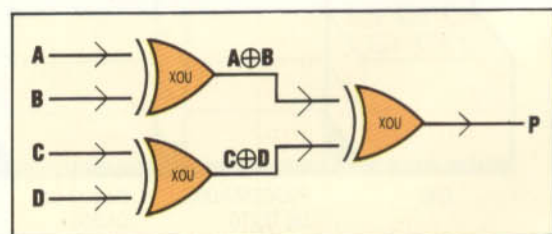
$$P = (\overline{A \oplus B}).(C \oplus D) + (A \oplus B).(\overline{C \oplus D})$$

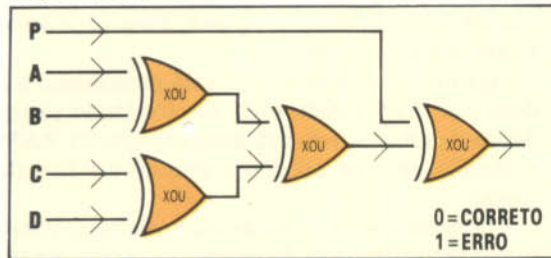
Considerando cada termo entre parênteses como uma entrada para uma porta XOU, a expressão se reduz a

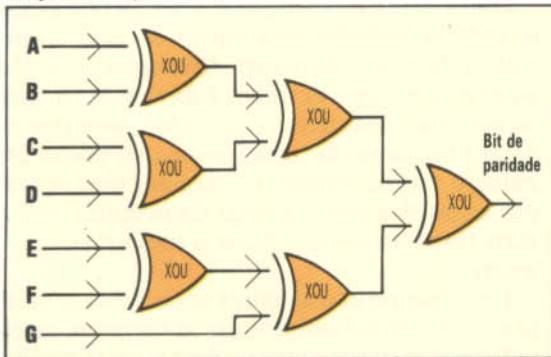
$$P = (A \oplus B) \oplus (C \oplus D)$$

e o circuito formado será uma "cascata" de portas XOU:

A	B	C	D	P
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Tabela de validação

Esse circuito funciona como verificador de paridade quando se acrescenta outra porta XOU para comparar os dois bits: o de paridade recebida é o gerado na extremidade receptora.

Código recebido de 5 bits

Na prática, a maioria dos computadores utiliza, para a transmissão de dados, o código ASCII, de 8 bits (7 de informação e 1 de paridade par). É fácil imaginar um gerador de bits de paridade para códigos ASCII:

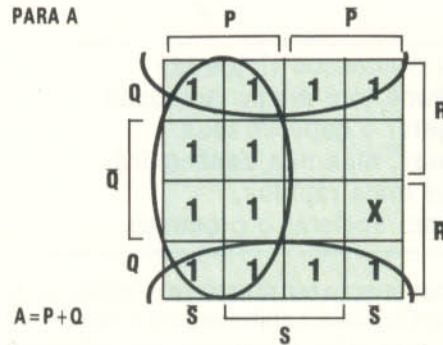
Código de informação de 7 bits

Decodificador de prioridade

Muitos computadores utilizam sistemas de "interrupção com prioridade" para controlar o fluxo de dados de [illegible] para os periféricos. Nesses sistemas, o periférico emite um sinal que interrompe a atividade da CPU. Quando ocorrem sinais simultâneos, segue-se uma ordem de prioridade, a fim de que a CPU "atenda" o periférico mais importante em primeiro lugar. O decodificador de prioridade que projetaremos unirá quatro periféricos num circuito capaz de identificar seus sinais e acionar um sistema de prioridade quando emitidos ao mesmo tempo.

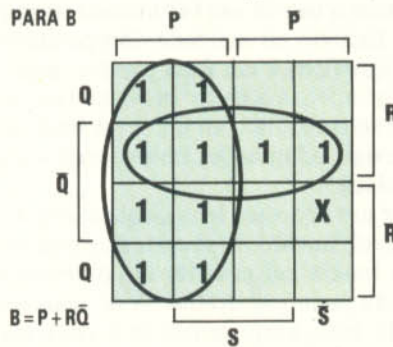
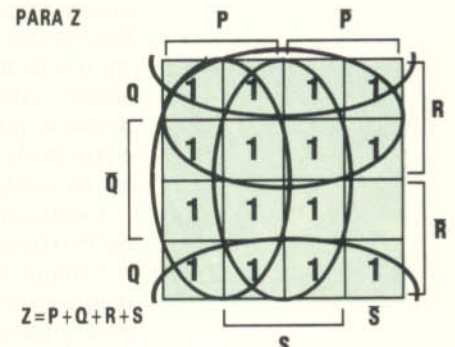
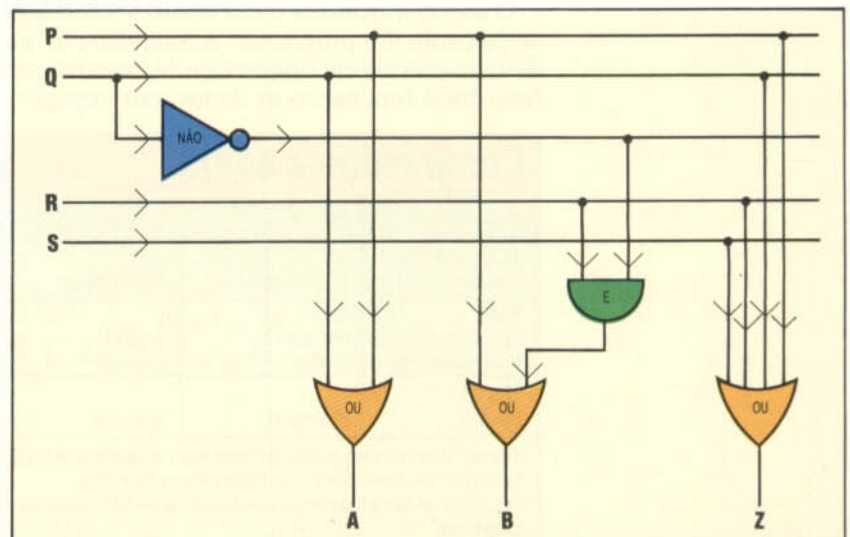
No caso de quatro periféricos, são necessárias duas linhas de saída e uma terceira para requisitar a interrupção. Considere os quatro periféricos P, Q, R e S, tendo P a prioridade máxima e S a mínima. A e B são linhas de saída para identificar o periférico. Z requisita a interrupção. A tabela de validação para o decodificador pode ser construída com o uso de um X para condições que "não interessam" ao decodificador:

P	Q	R	S	A	B	Z
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

Para compreender essa tabela, examine a última linha. No caso, P indica uma interrupção e, como se trata do periférico com prioridade máxima, não importa se os outros também estão sinalizando.

As três linhas de saída do circuito devem ser analisadas independentemente. Começando por A, o mapa-k será

O caso que "não interessa" do lado da saída A está representado por X no mapa-k, mas os que "não interessam" do lado da entrada são manipulados de modo diferente. Vejamos a situação em que P é 1 e Q, R e S são condições que "não interessam". Devemos preencher todos os quadros no mapa-k em que P é 1 — já há oito casos ao todo. Pelo mapa-k, obtém-se a expressão simplificada

$$A = P + Q$$

De modo semelhante, para B e Z, os mapas-k são

Usando essas três expressões, chegamos a este desenho de circuito:

MÉTODOS DE OTIMIZAÇÃO

Com os processos vistos até agora, você terá maior facilidade em projetar e depurar seus programas, mas não conseguirá torná-los mais rápidos. Veja como acelerar o processo.

Uma programação estruturada e o recurso a fluxogramas facilitam a elaboração e o uso de programas, mas não aumentam sua eficiência. Muitas vezes, para que estes sejam executados com mais rapidez e menos memória, é necessário sacrificar a clareza dos métodos utilizados. Então, sempre que otimizamos um bloco de programa de modo a torná-lo mais rápido, ele se torna mais difícil de ler, entender e corrigir.

A lenta execução inerente a linguagens interpretadas, como o BASIC, acarreta situações em que os programas rodam num ritmo inaceitável. E o modo mais eficiente de acelerar um programa em BASIC é compilá-lo. No entanto, poucos micros admitem o uso de um verdadeiro compilador BASIC. Existem no mercado compiladores em discos magnéticos e em fitas cassete, mas a maioria admite apenas o BASIC INTEGER (versão do BASIC que só trabalha com números inteiros) e isso pode exigir adaptações no programa antes da compilação.

Compilar é um processo lento, sobretudo durante o desenvolvimento do programa e quando o sistema se baseia em cassete. O compilador ocupa parte da memória destinada ao usuário; então, quanto mais abrangentes seus recursos, mais bytes da RAM ele ocupará na área destinada ao programa a ser compilado.

O acesso a arquivos reduz muito a velocidade de execução dos programas. Assim, entre os que lêem ou gravam em disquete ou fita cassete com freqüência (um banco de dados, por exemplo), a lentidão é inevitável. Num arquivo de acesso aleatório de um disquete, a recuperação de um registro demora, em média, 1/4 de segundo. Em arquivos seqüenciais, demora ainda mais (conforme o comprimento do arquivo). Com fitas cassete, o problema se agrava, pois os acessos ficam ainda mais lentos.'

Compiladores BASIC

Para micros	Fabricante	Gravado em
TK 85 e CP 200 Mcoder	Intercomputer	Fita cassete
Apple Task Compiler Hidden Compiler	 Microsoft Hidden	 Disquete Disquete
TRS-80 Bascom	Microsoft	Disquete

Existem várias versões do Mcoder. Uma das mais aperfeiçoadas é citada no livro *Super BASIC TK*, da Editora Aleph. Os outros compiladores são encontrados no mercado ou em bibliotecas de programas.

Quando essas demoras causam problemas, podemos reduzir o número de acessos lendo vários dados de uma só vez, armazenando-os na RAM e atualizando os arquivos apenas no fim da sessão.

Em geral, os programas interativos apresentam o inconveniente de deixar o usuário olhando para uma tela inerte por vários segundos. Uma solução parcial consiste em reorganizar o programa de modo que os arquivos sejam lidos e gravados enquanto o usuário está ocupado com outra coisa (lendo instruções no vídeo, por exemplo).

Outra causa da lentidão é a aritmética usada quando se trabalha com números reais — aritmética de ponto flutuante. (Números reais são aqueles com casas decimais à direita da vírgula; os inteiros — integers — não têm parte fracionária.) Por causa da parte fracionária, buscar um número real na memória e executar uma operação aritmética com ele exige da máquina muito mais trabalho do que fazer o mesmo com um integer.

Em programas com muitas operações aritméticas, convém substituir todas as variáveis envolvidas por variáveis integer, desde que o micro e a versão do BASIC utilizados permitam essa alteração. Dessa forma, consegue-se uma economia de até 20% no tempo de execução de programas sem muitas operações numéricas. Se o programa recebe, como dados de entrada, números previamente processados, consegue-se reduzir o tempo em até 50%.

Projetar um algoritmo mais rápido está entre as melhores maneiras de acelerar um programa. Procure vários algoritmos antes de se decidir por um. Muitos são publicados em livros e em revistas especializadas em computação.

Inventar algoritmos é questão de criatividade e inspiração. A maioria das versões do BASIC dispõe de várias funções eficientes e rápidas, como: INSTR (abreviatura de in string), que procura um substring dentro de uma variável instring; SGN (abreviatura de signal), que obtém o sinal de um número (−1, se negativo; 0, se nulo; e +1, se positivo); e LOG (abreviatura de logaritmo), que obtém o logaritmo decimal de um número. Às vezes vale a pena checar novamente o manual da máquina e ver quais as funções disponíveis, antes de se dar ao trabalho de reescrevê-las em sua própria versão.

Funções definidas pelo usuário, implementadas com o comando DEF FN (disponível em quase todas as versões do BASIC), também são executadas com rapidez. Esse comando é mais útil em programas com uma seqüência repetida de manipulação de strings, onde ele pode subs-

tituir uma chamada de sub-rotina. As rotinas se tornam mais rápidas quando escritas em linguagem de máquina. Isso porque as linguagens interpretadas traduzem cada uma das linhas de um programa para o ASSEMBLY à medida que vão sendo encontradas durante a execução do programa — processo bem pouco eficiente.

Codificar instruções em linguagem de máquina evita esse processo de tradução. Mas escrever um programa em linguagem ASSEMBLY é muito mais difícil do que em BASIC, e o custo (em tempo e esforço) da programação nem sempre compensa a economia no tempo de execução. Por outro lado, alguns programas — os que usam gráficos animados, por exemplo — não funcionariam como planejado se fossem escritos somente em BASIC.

Há muitas outras maneiras de conseguir economias menores na velocidade de processamento. Em alguns micros, vale a pena usar uma variável no lugar de um número (por exemplo, MAX em vez de 267,5) para obter um acesso mais rápido aos valores, sobretudo em loops. Empregue letras diferentes para iniciar os nomes das variáveis; se necessário, utilize todo o alfabeto. Procure usar linhas com várias instruções, quando possível. Se o interpretador permitir, elimine dos loops FOR-NEXT a variável contadora da instrução final, isto é, use apenas NEXT em vez de NEXT e o nome da variável contadora. Dentro de um loop, evite calcular o mesmo valor em cada passagem. Em vez disso, calcule o valor fora do loop e incorpore-o como uma variável.

Economizando espaço

A aritmética usada para números integer, além de tempo, economiza espaço. Empregam-se 4 ou 5 bytes para armazenar um número real e apenas 2 para um integer. Isso representa bastante economia, sobretudo quando são usadas grandes matrizes.

Alguns aumentos na velocidade de um programa também economizam espaço. É o que ocorre quando se usam funções residentes ou definidas pelo usuário, se escreve em linguagem ASSEMBLY ou se recorre a linhas com várias instruções. Por outro lado, a compilação tende a aumentar o tamanho de programas pequenos, só economizando espaços em programas grandes.

A eliminação das linhas REM economiza um bom espaço, e o uso de strings de texto mais curtos para mensagens também ajuda. Colocar grandes blocos de texto em arquivos armazenados fora do programa exclui esses blocos quando não necessários (as instruções e os textos de auxílio ao usuário são o que mais ocupa espaço). Remova os espaços dispensáveis de cada linha, use números de linhas pequenos e nomes mais curtos para as variáveis. Se uma matriz precisa ser dimensionada, mas você desconhece seu tamanho exato, não arrisque um número qualquer. Em vez disso, espere até obter a informação necessária e, então, dimensione-a com uma variável, como neste exemplo:

```
10 INPUT "QUANTOS CASOS HA NESSA
   CATEGORIA?";CASOS%
DIM MATRIZ%(CASOS%)
```

Isso se chama "dimensionamento dinâmico" e é um recurso oferecido pelo BASIC, mas não por muitas outras linguagens. Utilize-o ao máximo.

Outra técnica usada em alguns micros é o aumento da área na RAM para o programa. Pode-se fazer isso por meio de comandos como o HIMEM (do Apple). Em geral esses comandos mudam a área da RAM disponível para programas em BASIC e variáveis. Tal mudança serve para armazenar programas em código de máquina num lugar seguro, onde nada será gravado sobre eles. Mas o mesmo comando permite acessar espaço extra da área reservada à memória de vídeo, constituindo uma boa maneira de se conseguir uns quilobytes a mais de RAM quando o que está aparecendo na tela não é importante. Caso não seja possível alterar a área reservada para programas, use a memória de vídeo diretamente, por meio dos comandos PEEK e POKE, nas localizações da memória reservada para ela.

Quando tudo falha e o programa não cabe no espaço disponível, algumas versões do BASIC dispõem de um comando (CHAIN) que permite a um programa passar o controle para outro. Outras versões empregam o comando COMMON, que transfere determinadas variáveis e seus valores atuais para o próximo programa. Quando existente nos micros, o CHAIN — em geral um comando muito simples — possibilita a transferência de todas as variáveis (ou de nenhuma) do primeiro programa para o segundo.

Em programas elaborados de maneira estruturada, cada rotina deve ser escrita e testada independentemente. Pode-se também cronometrar a execução de cada uma. Um programa medidor de tempo que roda no TK 90X é o seguinte:

```
100  REM  Use esta secao para estabelecer
110  REM  variáveis que serao utilizadas pela
120  REM  rotina a ser testada (nao
130  REM  se esqueça de dimensionar
140  REM  matrizes e preenche-las com dados
150  REM  realistas). Uma variavel do sistema
          chamada TVCOUNT e usada para
             medir o tempo.
160  POKE  23672,0
170  POKE  23673,0
180  POKE  23674,0
190  LET  inicio = PEEK 23672 + 256*PEEK
          23673 + 65536*PEEK 23674
200  GOSUB  2000: REM a rotina a ser
            cronometrada deve ser
            chamada aqui.
210  LET  fim = PEEK 23672 + 256*PEEK
          23673 + 65536*PEEK 23674
220  PRINT  "A execucao demorou "; (fim-inicio)
            /60," segundos. "
2000  FOR  a = 1 TO 10000: NEXT a
2010  RETURN
```

Com essa rotina é possível experimentar diferentes algoritmos e maneiras de aumentar a velocidade.

LÓGICA EMPRESARIAL

Seguindo a tendência do mercado internacional, em 1985 a Prológica lançou seu micro de 16 bits, o SP 16, compatível com o IBM PC.

Fruto da ousadia de dois amigos entusiastas da eletrônica, a Prológica tornou-se, em nove anos, uma das maiores e mais bem-sucedidas empresas do setor de micros pessoais e profissionais.

Uma modesta firma de distribuição de produtos eletrônicos, na tradicional rua Aurora, no centro velho de São Paulo, foi palco, em março de 1976, de importante decisão. Dois amigos, unidos pelo prazer de experimentar ludicamente equipamentos eletrônicos, resolveram comercializar o protótipo da máquina contábil que acabavam de montar. Seis meses depois, Leonardo Bellonzi e Joseph Blumenfeld lançavam sua engenhoca no mercado, passando de distribuidores a produtores de equipamentos eletrônicos.

A tentativa revelou-se um sucesso: em dois anos venderam-se mais de 5.000 unidades, confirmando a possibilidade comercial da nova associação. Assim nasceu a empresa que, a partir de 1980, ocuparia destacada posição entre as produtoras nacionais da área de informática — a Prológica Indústria e Comércio de Microcomputadores Ltda.

Nove anos depois, em 1985, a Prológica ocupava o terceiro lugar na classificação das empresas nacionais do setor, empregando mais de 1.500 funcionários. Esse desempenho invejável deve-se em grande parte à filosofia, adotada desde sua fundação, de vender não apenas equipamentos mas, sobretudo, soluções. Todos os produtos foram desenvolvidos a partir de pesquisas sobre as necessidades de seus clientes potenciais.

A Prológica conta hoje com extensa rede de pontos de venda e com programas de treinamento oferecidos a clientes e usuários.

Depois do primeiro sistema contábil, lançado no final da década de 70, a empresa concentrou suas atividades na produção de microcomputadores pessoais (40%), sistemas profissionais (40%) e periféricos (20%). A linha de micros inclui desde modelos mais simples, como o CP 200 e o CP 300, até equipamentos com recursos avançados, como o CP 400 Color e o carro-chefe da linha, o CP 500. No setor dos equipamentos profissionais, a Prológica produz o Sistema 700 e o Super 700, ambos de 8 bits, e o SP 16, um micro modular de 16 bits compatível com o IBM PC. Impressoras matriciais, unidades de disco rígido tipo Winchester e unidades de disco flexível completam sua linha de produtos.

Acompanhar as modificações de um mercado dinâmico como o da informática implica investimento constante na criação de novos produtos. Por isso, trezentos engenheiros e técnicos da Prológica dedicam-se exclusivamente à pesquisa tecnológica, que consome cerca de 10% do faturamento da empresa.

Com a diversificação dos produtos, a empresa vem ampliando cada vez mais suas instalações. Desta linha de montagem saem três modelos de impressoras matriciais: P-500, P-720 e P-740.

TÚNEL DO TEMPO

Na escola, um micro pode criar meios ambientes, sem limitações de tempo e espaço. Seu enorme potencial para a educação aumentou ainda mais com os jogos de simulação e aventura.

Existe alguma razão para o aprendizado não ser tão divertido quanto os jogos? Todas as crianças — e a maioria dos professores — responderão negativamente à pergunta. O uso dos jogos (em especial dos dramáticos, com desempenho de papéis) adquire cada vez mais popularidade, acompanhando o desenvolvimento das técnicas de dramatização na educação e da teoria dos jogos na psicologia. Tais desenvolvimentos deram ainda mais crédito ao potencial dos computadores nas escolas.

Os videojogos de aventura e simulação do tipo Invasores do Espaço e outros mais recentes representam as duas categorias predominantes desse tipo de diversão. E, à medida que o software se desenvolve, torna-se mais difícil distinguir as diferenças entre elas; no entanto, a classificação persiste.

Apesar de sua popularidade, os jogos do gênero guerra espacial tiveram pouco impacto sobre o software educacional. Ao escreverem o livro *Mind at play, the psychology of video games (O pensamento em jogo, a psicologia dos videojogos),* Elizabeth e Geoffrey Loftus perguntaram a vários psicólogos qual a eficácia educacional desses jogos. As respostas não foram encorajadoras. O valor de entretenimento do tipo "acabar-com-eles" ultrapassa em muito qualquer valor educativo. No entanto, a pesquisa mostrou que, devido à coordenação e aos reflexos exigidos, esse tipo de jogo pode ter efeitos terapêuticos sobre os portadores de vários tipos de desequilíbrio mental e de lesões no cérebro.

Houve tentativas de reforçar o conteúdo educacional desses jogos: ao soletrar corretamente uma palavra, por exemplo, a criança abatia os invasores. Todavia, comprovou-se que os jogos modificados eram menos divertidos que os originais — e o benefício educacional era, na melhor das hipóteses, duvidoso.

Aventuras na escola
Os jogos educacionais de aventura e simulação tornam possível à criança explorar, do interior da sala de aula, situações de outra forma inacessíveis ou mesmo potencialmente perigosas. A experiência tem mostrado que esses jogos, longe de ser uma perda de tempo, podem acrescentar muito ao processo de aprendizado.

Viagem no tempo
O Projeto Simulação-Banco de Dados usa o computador como ferramenta suplementar, destinada a atuar juntamente com outras atividades de aprendizado. O pacote de programas divide-se em três partes: um jogo de aventuras, um conjunto de arquivos de texto que fornece informações básicas e um banco de dados contendo outras referências e fontes. Neste exemplo, a turma de escolares foi dividida em seis grupos e cada um assumiu o papel de um personagem em viagem pela Escócia do final do século XIX. As atividades decorrentes do uso do banco de dados incluíram redação criativa, cartografia, dramatização e excursões pela Escócia contemporânea.

Em contrapartida, os jogos de simulação e de aventura apresentam grande potencial como alternativas educacionais. Os primeiros usam o computador para criar instrumentais de aprendizagem até então inacessíveis à criança; e os de aventura reproduzem cenários fantásticos, sons e efeitos especiais das fábulas e da ficção científica. Esses princípios já são aproveitados nos programas de treinamento industrial, e hoje se empregam muitos deles em escolas.

Ballooning (Uma Viagem de Balão) é um jogo de simulação típico, criado na Grã-Bretanha. Na parte inferior da tela há um painel de instrumentos, com altímetro, marcador de combustível, de temperatura, e um indicador da velocidade de descida ou de ascensão, que reage conforme se acionam o queimador e a válvula de gás. Quando o balão desce a certa altura, o terreno torna-se visível; caso haja necessidade de abastecimento, existem vários postos terrestres para os quais o balão deve ser conduzido.

Vantagens educacionais

Um computador pode apresentar em detalhe a estrutura física de qualquer experimento ou exercício. As técnicas de simulação permitem ao aluno concentrar-se apenas na situação em desenvolvimento, no que está acontecendo e no porquê. O tempo deixou de ser um obstáculo à finalização, em sala de aula, de projetos complexos e longos: experiências que exigiram dias no laboratório surgem simuladas na tela em segundos. Um modelo em computador simplifica sistemas que seriam complexos demais para o entendimento dos estudantes, e gráficos podem realçar funções pertinentes, por meio de dados representativos, a partir de estágios relevantes no processo.

Além disso, como a simulação pode se repetir sob várias condições, não há limite para o número de experimentos à disposição dos estudantes. Estes controlam as variáveis implementadas e os rumos que o experimento toma, o que lhes proporciona melhor compreensão dos resultados e das conclusões.

Mas existem limites ao uso da simulação. Como o computador só pode armazenar um número predefinido de variáveis, há o risco de uma simulação apresentar visão por demais simplificada do tema. Assim, embora seja seu maior atributo, a capacidade de representar situações reais é também sua maior falha.

Um programa nos moldes do Ballooning, com número fixo de variáveis, atinge seu maior benefício quando usado, num contexto mais amplo, como um entre muitos aspectos da ciência e da história da aeronáutica. Mas muitas vezes se ignora que a simulação não substitui a realidade. Sua força está em estimular o interesse por temas associados e não em representar uma extensão das técnicas de ensino programado.

Exemplo extremo de emprego questionável de computador é o programa que simula um circuito ligando lâmpadas e baterias de várias maneiras. Ao que parece, foi desenvolvido para uma escola capaz de adquirir um computador caro, mas que, por algum motivo misterioso, não podia comprar pilhas e lâmpadas.

Criando micromundos

Nos últimos anos, livros e filmes como *O senhor dos anéis, 2001: Uma odisséia no espaço* e *Conan, o bárbaro,* conquistaram enorme popula-

ridade. Nada estranho, visto que a maioria das pessoas — crianças e adultos — gosta de abandonar-se à fantasia. E não é por outro motivo que os jogos de aventura se tornaram enorme sucesso comercial, além de valorizarem o desempenho de papéis dramáticos — o que os especialistas chamam de "role playing" — na terapia e na educação.

A natureza interativa do computador faz dele uma ferramenta ideal para os jogos de fantasia e de aventura. Suas possibilidades gráficas e sonoras podem dar "realidade" a um mundo fantástico ou a uma época do passado. E os educadores logo se apossaram dessa capacidade de criar "micromundos", vendo-os como um salto qualitativo nas práticas de ensino.

Aparentemente, não haveria melhor maneira de uma criança aprender história do que viajando ao passado e vivenciando uma aventura. Mas esse "viajante no tempo" necessitaria de informações para sobreviver; assim, o programa deveria conter um banco de dados que servisse de fonte de consulta durante o jogo.

O Projeto Simulação-Banco de Dados (Simulation Database Project) foi elaborado por Allan Martin no St. Andrews College of Education, de Glasgow, "para explorar a forma pela qual os computadores podem contribuir de modo significativo para o trabalho na escola primária, criando um novo tipo de objeto pedagógico baseado no computador".

A estrutura da simulação lembra bastante um livro de aventuras: a criança ingressa num mundo histórico, viaja por diferentes locais e enfrenta várias situações. Os arquivos de texto no banco de dados estão relacionados a cada local e são acessíveis a qualquer momento. Um segundo banco de dados contém detalhadas referências ao local ou assunto e pode ser pesquisado para informações complementares em livros, filmes e outros meios de comunicação. A maior parte do aprendizado corre independentemente do computador, que é apenas um entre vários recursos educacionais.

Para melhor integração do programa a outras atividades em classe, o software inclui um manual, um mapa da área, cartas, ilustrações e outros materiais de consulta. Os programas podem ser ajustados a condições locais e são escritos em LOGO — linguagem utilizada por um número crescente de escolas brasileiras e que, segundo os especialistas, facilita o desenvolvimento do raciocínio lógico e desperta a criatividade infantil, constituindo uma porta de entrada ideal para o mundo dos computadores.

Um bom exemplo de banco de dados simulado tem por tema a Escócia da década de 1880. O programa foi usado com uma turma dividida em seis grupos e cada qual assumiu a identidade de um personagem fictício — mas autêntico — representativo de uma camada social: aristocratas, industriais, religiosos, operários, empregados domésticos etc. Cada grupo recebeu um roteiro e era obrigado a viajar pela Escócia. Após planejar a viagem com o mapa e chegar ao primeiro local, o grupo-personagem deparava-se com vários incidentes. Depois de cada fase de movimento, as crianças, usando dois bancos de dados e outras fontes de consulta fornecidas pelo professor, investigavam o local. As atividades derivadas dessas investigações incluíam cartografia, elaboração de diários e reportagens para jornais, desenhos e pinturas do ambiente. Além disso, as crianças produziram peças, gravaram "entrevistas" com personagens da época, visitaram as áreas abrangidas pelo jogo e convidaram oradores alheios à escola para realizarem conferências sobre a Escócia.

O Projeto Simulação-Banco de Dados exemplifica como um videojogo pode ser usado para interessar alunos e estimular o trabalho em equipe. Dois outros pacotes deste mesmo projeto começaram a ser elaborados em 1984: Palestina no Tempo de Cristo, empregado em cursos histórico-religiosos, e Os Rios, subsídio aos estudos do meio ambiente.

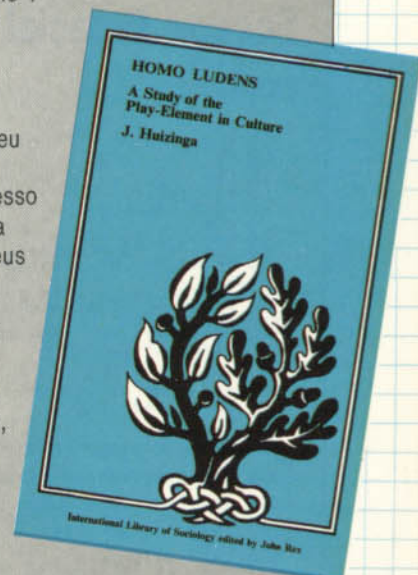
Essas simulações empregando as excepcionais qualidades do computador produzem um novo e fascinante recurso educacional. O micro torna-se um meio pelo qual a criança pode adquirir uma experiência valiosa, entrando em contato com ambientes e épocas históricas até então inacessíveis — outros mundos.

Homo ludens

Um programa na tevê britânica mostrou um grupo de crianças desenhando uma tartaruga mecânica guiada pela linguagem LOGO no chão da sala de aula. O locutor observou aos telespectadores que eles não deviam se iludir com o caráter improvisado do exercício, afirmando: "De fato, aqui está ocorrendo aprendizado real".

Tais palavras certamente perturbaram os pais e professores que acreditam na tradicional diferenciação entre "brincadeira" e "estudo sério". A verdade, porém, é que os limites entre essas categorias tendem a diluir-se — e muitos acreditam que o aprendizado "real" ocorre somente através dos jogos.

Há uma base para esse ponto de vista. Em seu *Homo ludens*, lançado em 1944, o pensador holandês J. Huizinga afirma que, além de processo de aprendizado, o jogo tem valor substancial na cultura humana por si mesmo. Eis alguns de seus argumentos:

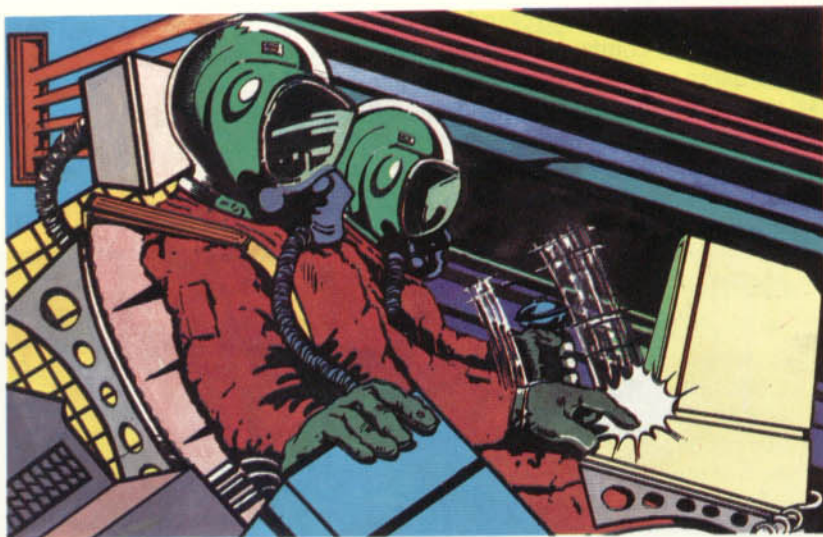
- Jogar é uma função significativa, dotada de sentido.
- O jogo é voluntário e instintivo. As crianças brincam para aprender sobre o mundo "adulto", mas continuam a fazê-lo por ser divertido.
- A civilização contemporânea sofre de "formalização de jogos": quanto mais estruturados, menos são jogos.

TESTE DE REFLEXOS

Crianças e adolescentes conseguem melhor desempenho em jogos de ação, onde uma resposta rápida é tudo o que importa. O jogo aqui apresentado permite que você meça seu tempo de reação desviando-se de asteróides.

Todos gostamos de pensar que nossos reflexos são ótimos, que nossas reações aos eventos acontecem instantaneamente. No entanto, reagindo a um estímulo, a resposta mais rápida de que somos capazes demora cerca de um terço de segundo. Parece pouco, mas tal defasagem pode ser fatal. Um carro em alta velocidade, por exemplo, percorre quase 10 m nesse tempo.

Os tempos de reação variam muito de uma pessoa para outra, e álcool, cansaço ou doença exercem efeito desfavorável sobre os resultados. O programa Socorro Espacial permite testar os reflexos de qualquer pessoa e, para tornar os tempos mais precisos, considera a média obtida em cinco tentativas.

Neste jogo, você pilota uma espaçonave e precisa levar com urgência sua carga de medicamentos a certa colônia instalada num cinturão de asteróides. A velocidade é vital para que o socorro chegue a tempo, mas a nave deve ser desviada sempre que houver perigo de colisão. Acionando a barra de espaço, você pára a espaçonave, evitando bater nos asteróides. O jogo mostra o tempo transcorrido entre o momento em que o asteróide aparece na tela e o instante em que se pressionou a barra de espaço. Após cinco paradas da nave, a média do tempo de reação é calculada.

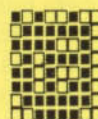
Para se certificar de que o jogo vem transcorrendo com seriedade, o programa verifica se a barra de espaço não está sendo pressionada continuamente. Se estiver, os motores serão desligados e soará um alarme. Quando os asteróides surgem no vídeo, o radar da nave emite um som. Se quiser dificultar o exercício, mude o programa de modo a receber um alerta auditivo ou visual, mas não os dois.

Socorro Espacial

TK 85

```
  10 REM ▞▜ PARA O TK85 ▛▚
  20 LET M=0
  30 LET C=0
  40 LET V=0
  50 DIM R(5)
  60 POKE 16418,0
  70 CLS
  80 FOR G=1 TO 5
  90 FOR P=1 TO RND*30+10
 100 PRINT AT INT (RND*22),INT (
RND*32);"."
 110 IF INKEY$="" THEN GOTO 150
 120 PRINT AT 21,0;"AGUARDE O AS
TEROIDE"
 130 IF INKEY$<>"" THEN GOTO 130
 140 PRINT AT 21,0;"
       "
 150 NEXT P
 160 PRINT AT 11,16;"■"
 170 POKE 16436,255
 180 FAST
 190 LET P=P+1
 200 IF INKEY$<>"P" THEN GOTO 18
0
 210 LET R(G)=10+10*(P-1110)/110
0
 220 PRINT AT 22,0;"VOCE DEMOROU
 ";INT (R(G)*100)/100,"SEGUNDOS
PARA PARAR OS MOTORES."
 230 SLOW
 240 LET M=M+R(G)
 250 FOR L=1 TO 150
 260 NEXT L
 270 CLS
 280 NEXT G
 290 LET A=M/5
 300 CLS
 310 PRINT "APOS 5 TENTATIVAS, S
UA MEDIA DE TEMPO DE REACAO FOI
DE ";INT (A*100)/100,"SEGUNDOS"
 320 FOR G=1 TO 5
 330 LET V=V+ABS (R(G)-A)
 340 NEXT G
 350 PRINT
 360 PRINT "SEU TEMPO DE REACAO
VARIOU ",INT (V*20/A);"%."
 370 PRINT
 380 PRINT "VOCE QUER TENTAR NOV
AMENTE?(S/N)"
 390 LET R$=INKEY$
 400 IF R$="S" THEN RUN
 410 IF R$<>"N" THEN GOTO 390
```

TK 90X

```
  10 REM ▞▜ PARA O TK90X ▛▚
  20 INK 7: PAPER 0: BORDER 0
  30 LET M=0: LET A=0: LET V=0:
DIM R(5): CLS
  40 FOR G=1 TO 5
  50 FOR P=1 TO RND*150+50
  60 PLOT INT (RND*250),INT (RND
*170)
  70 IF INKEY$<>" " THEN GOTO 90
  80 SOUND .1,-3: GOTO 70
  90 NEXT P
 100 FOR A=1 TO 4
 110 PRINT AT RND*20,RND*30;"@"
 120 NEXT A
 130 SOUND .05,15: LET C=0
 140 LET C=C+1: LET A$=INKEY$
 150 IF A$<>" " THEN GOTO 140
 160 LET R(G)=C/66+.05
 170 PRINT AT 19,0;"VOCE DEMOROU
 ";INT (R(G)*100)/100,"SEGUNDOS
PARA PARAR OS MOTORES."
 180 LET M=M+R(G)
 190 FOR J=1 TO 300: NEXT J
 200 NEXT G
 210 LET A=M/5: CLS
 220 PRINT "APOS 5 TENTATIVAS, S
UA MEDIA DE TEMPO DE REACAO FOI
DE ";INT (A*100)/100,"SEGUNDOS."
 230 FOR G=1 TO 5: LET V=V+ABS (
R(G)-A)
 240 NEXT G
 250 PRINT : PRINT "SEU TEMPO DE
REACAO VARIOU ",INT (V*20/A);"%
."
 260 PRINT : PRINT "VOCE QUER TE
NTAR NOVAMENTE?(S/N)"
 270 LET R$=INKEY$: IF R$="s" TH
EN GOTO 30
 280 IF R$<>"n" THEN GOTO 270
 290 INK 0: PAPER 7: BORDER 7: C
LS
```

Obs.: Este e' o caractere usado na linha 110 .

O MELHOR DE DOIS MUNDOS

Um televisor combinado com monitor próprio para micros permite alta qualidade de imagem e o uso de recursos sonoros, além de receber transmissões normais de tevê.

Talvez você possua um televisor-monitor sem ter se dado conta disso, pois muitos dos novos aparelhos receptores de tevê vêm equipados com saídas para videocassete adequadas também a micros.

Os problemas associados ao uso de televisores comuns como vídeos de computadores decorrem da maneira como eles recebem os sinais. As emissoras transmitem seus programas sob a forma de ondas de rádio, que são captadas pela antena e convertidas em imagens.

Um micro pode simular esse procedimento enviando, através de um modulador, sinais que depois, dentro do receptor, serão de novo alterados. Sofrendo duas modificações — no modulador e no receptor de televisão —, o sinal perde definição. O monitor, por sua vez, não necessita de modulação, recebendo o sinal da própria imagem, exibindo-a com nitidez.

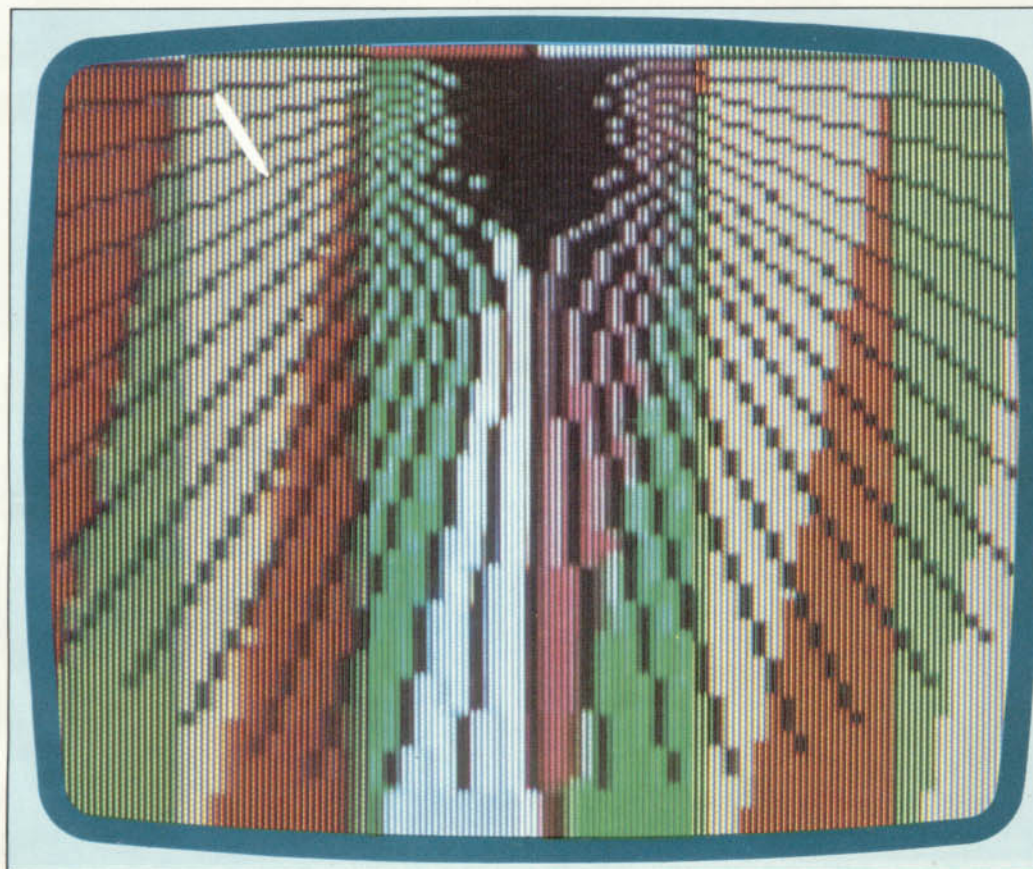
Há televisores adaptados para se comportar como monitores e outros já projetados com entrada para videocassete. Estes se mostram mais eficientes, de vez que os primeiros são adequados a valores de tolerância bem mais amplos que os desenvolvidos para funcionarem como monitores.

Os televisores-monitores têm entrada de vídeo composto (indicada como "vídeo" ou "audiovisual"). Os diagramas e manuais que acompanham o aparelho mostram como ligar a uma dessas saídas os micros compatíveis. Em seguida, deve-se sintonizar o televisor do mesmo modo como se selecionaria um canal de emissora.

A principal vantagem do televisor-monitor está no recurso sonoro, bem superior ao de um monitor padrão. A maior parte dos microcomputadores — sobretudo os modelos da Microdigital e da Prológica — conta com televisores para a produção de efeitos sonoros. Um monitor padrão não possui recursos de som, ao passo que os televisores-monitores se valem de alto-falantes e amplificadores embutidos.

O desempenho superior do combinado televisor-monitor acabará fazendo desses aparelhos os mais procurados pelos usuários de micros.

Sinais e imagens

Existem três tipos principais de sinais produzidos por computadores. Todos os micros produzem sinais compatíveis com os de televisores, mas isso freqüentemente significa imagem insatisfatória. A maioria dos computadores também produz sinais para monitores, que fornecem melhor imagem mas são muito caros.
Os televisores-monitores combinam a qualidade do monitor com a capacidade de receber imagens de televisão. A diferença fundamental na qualidade entre televisores e monitores está no tipo de sinal com que trabalham. Estas três imagens foram produzidas no mesmo televisor-monitor, com três tipos diferentes de sinal: de tevê, de vídeo composto e de RGB (Red, Green, Blue).
O sinal de televisão *(no centro da tela)* fornece a imagem menos satisfatória; o de vídeo composto *(à esquerda)* é um pouco melhor; e o sinal RGB *(à direita)* dá os melhores resultados.

Maior nitidez
O televisor-monitor permite que seus videojogos sejam vistos com nitidez muito superior à conseguida num televisor comum.

Alta definição
A boa definição, na tela, de softwares gráficos, textos e desenhos torna menos cansativo o trabalho com programas extensos.

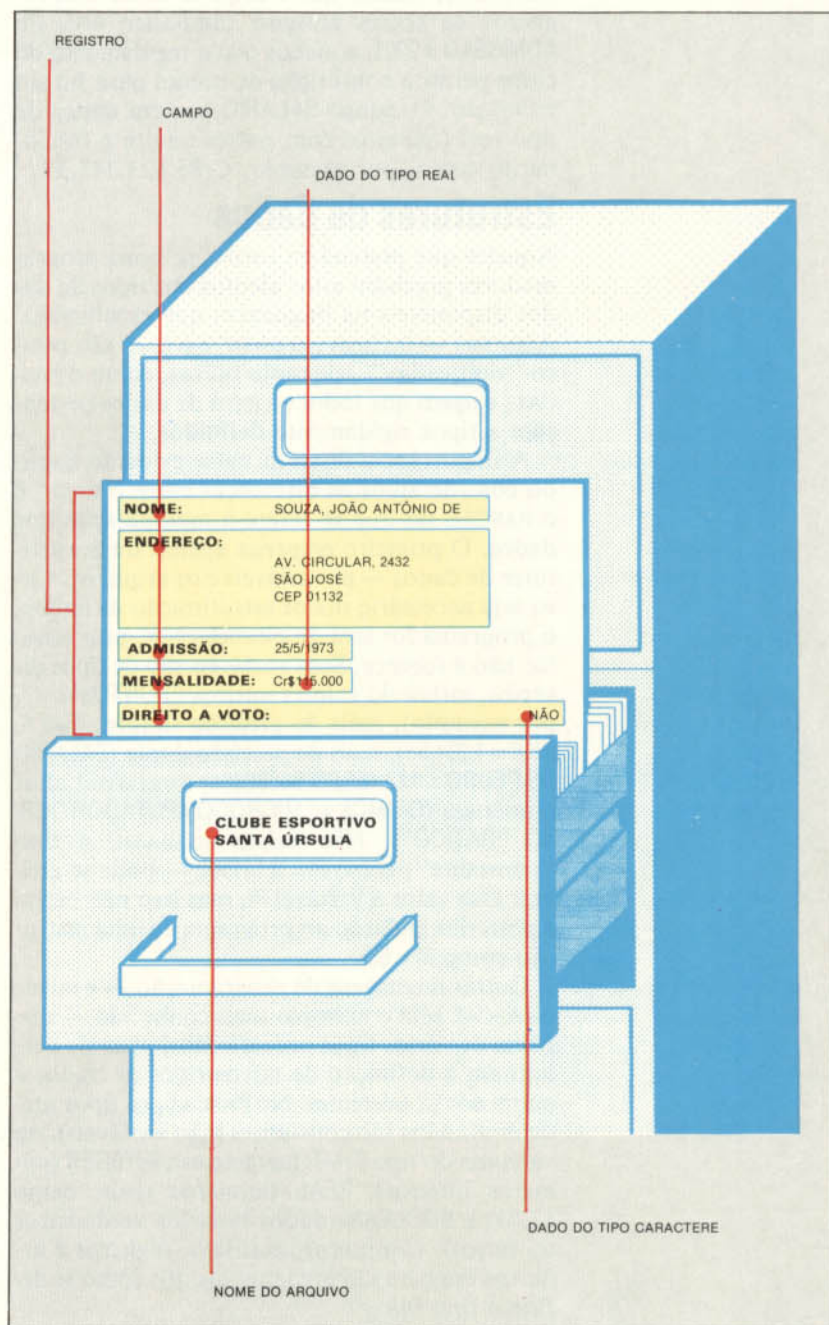
ORGANIZAÇÃO DE DADOS

Quando bem elaborado, um banco de dados permite a organização e manipulação de informações, facilitando o acesso a todos os itens armazenados. Vamos ver como funciona esse recurso indispensável.

Muitas pessoas e instituições vêem-se obrigadas atualmente a manipular consideráveis volumes de informações estruturadas — secretários de clubes, bibliotecários, órgãos governamentais, corretores de imóveis, hospitais, escolas, empresas etc. E a tarefa de organização e atualização dessas informações fica extremamente facilitada pelo emprego de computadores e programas gerenciadores de bancos de dados. Antes, porém, de examinarmos as técnicas desenvolvidas para a utilização desses programas, vamos a uma breve revisão dos conceitos fundamentais envolvidos.

Para os programadores, os dados dividem-se em estruturados e não estruturados. Estes últimos são unidades isoladas, em geral armazenadas em variáveis ou em constantes — por exemplo, TENTATIVAS := TENTATIVAS + 1 ou IF CERTO = 1 THEN PROCEDURE INCREMENTO. Nos dois exemplos, TENTATIVAS e CERTO são variáveis; no primeiro caso, o valor da variável está sendo alterado e, no segundo, está sendo testado (com possível desvio para uma sub-rotina). As variáveis TENTATIVAS e CERTO somente podem assumir um valor de cada vez.

Os programadores que utilizam o BASIC acostumaram-se a pensar nos dados apenas como não estruturados — ou seja, como variáveis —, ao contrário do que ocorre no cotidiano, onde, na maioria das vezes, agimos a partir de dados estruturados. Considere, por exemplo, seu grupo de amigos: certamente você tem uma "estrutura de dados" própria para manipular o conjunto de dados referente a eles. Afinal, conhece seu nome, idade, sexo e outros dados como profissão, salário, hábitos, endereço etc. Um conjunto de fatos inter-relacionados sobre uma pessoa, conceito ou objeto recebe o nome de "registro". Cada registro compõe-se de um ou mais campos (de tipos iguais ou diferentes), e o conjunto de todos os registros é um arquivo.

Um exemplo ajudará a entender isso melhor: imagine o secretário de um clube esportivo com 1.500 associados. Para manter um controle eficiente, ele utiliza um fichário — com fichas de todos os sócios. Cada uma contém informações (dados) com características (tipos) diversas, relativas a um único associado. As informações incluem, por exemplo, sobrenome, nome, sexo, idade, ano de admissão, mensalidades pagas, salário, número do sócio, e assim por diante.

Os dados aceitáveis para os campos SOBRENOME e NOME são necessariamente representados por cadeias de caracteres alfanuméricos — afinal, 1374662 não seria apropriado para o nome de uma pessoa. O item SEXO aceita um dado de

tipo booleano, que admite apenas um em dois valores binários. IDADE será um dado de tipo integer (ou seja, um número inteiro, visto que nenhum secretário necessita saber que determinado sócio tem 37,624 anos de idade). Os dados para o campo ANO DE ADMISSAO também serão valores integer, mas pertencentes a uma faixa limitada. Assim, ANO DE ADMISSAO = 1530 não seria adequado, mesmo que o arquivo contivesse registros de sócios antigos; tampouco ANO DE ADMISSAO = 2001, a menos que o regulamento do clube permita a inscrição de nomes para futura admissão. O campo SALARIO contém dados de tipo real (números com partes inteira e fracionária) como, por exemplo, Cr$5.823.342,37.

Estruturas de dados

Aqueles que pretendem tornar-se bons programadores precisam estar atentos aos tipos de dados disponíveis na linguagem que escolheram. Algumas — inclusive o BASIC e o C — são pouco "tipificadas", enquanto outras, como o PASCAL, exigem que todos os itens de dados pertençam a tipos rigidamente definidos.

A importância disso torna-se evidente quando consideramos as diferenças entre o BASIC e o PASCAL no que se refere à manipulação dos dados. O primeiro emprega apenas duas estruturas de dados — as variáveis e os arquivos. Caso seja necessário maior estruturação de dados, o programador terá de estabelecê-la, pois o BASIC não a fornece. Suas variáveis são de tipos diversos, incluindo valores inteiros (PONTOS% = 7, por exemplo), reais de precisão simples (SALARIO! = 1.234,56), reais de precisão dupla (SALARIO DE PEDRO = 123.456,66666666666) e variáveis alfanuméricas (OTIMO$ = "MICROCOMPUTADOR CURSO PRATICO"). Não existe no BASIC o tipo "constante", como PI = 3,141592 — pode-se atribuir esse valor à variável PI, mas isso não exclui a posterior inclusão no programa de uma instrução como PI = 6,2.

Outras linguagens de programação — e talvez o PASCAL seja o exemplo mais conhecido — dispõem de vários tipos predefinidos, além de permitirem a definição de novos tipos de dados a partir dos já existentes. No PASCAL, os tipos predeterminados são constantes (não variáveis), ou variáveis do tipo CHAR (caracteres), INTEGER (números inteiros), REAL (números reais, como 12,71) e BOOLEAN (dados binários verdadeiros ou falsos). Conjuntos, matrizes, registros e arquivos também são predefinidos. Eis como se define o tipo DIA:

```
TYPE
  DIA = (SEGUNDA, TERCA, QUARTA, QUINTA,
         SEXTA, SABADO, DOMINGO);
```

Qualquer variável do tipo DIA usada no programa assume somente um dos valores especificados; portanto, se definíssemos FOLGA como uma variável pertencente ao tipo DIA, então FOLGA := NOVEMBRO não seria válido, ao contrário de FOLGA := DOMINGO.

Denominam-se bancos de dados os conjuntos de informações inter-relacionadas e estruturadas. Por convenção, os dados são organizados num arquivo (identificado por um nome), o qual compõe-se de registros (indicados por números), que, por sua vez, contêm um ou mais campos. Estes, finalmente, podem incluir um ou mais tipos de dados.

Supondo que o banco de dados do clube possua a mesma estrutura de um fichário, o mais provável é que seu principal código de acesso esteja baseado na ordenação alfabética dos sobrenomes dos sócios. Quando há necessidade de encontrar os sócios de sobrenome Souza, o secretário do clube percorre o fichário até encontrar uma ficha com a letra S. Em seguida, diminui a velocidade de busca e passa a procurar os sobrenomes que comecem por SO, até achar os registros dos Souzas. No entanto, caso ele precisasse verificar quantas sócias existem no clube, a pesquisa seria certamente muito mais trabalhosa. E ainda mais difícil e demorada seria a tarefa de descobrir quantos sócios recebem salários acima de 10 milhões de cruzeiros, têm idade superior a 37 anos e mais de dois filhos.

Esse tipo de tarefa é precisamente o que um programa ou sistema gerenciador de banco de dados realiza em pouco tempo ao simples toque de algumas teclas do computador. Segundo uma definição rigorosa, o banco de dados corresponde às informações, enquanto o programa que possibilita a introdução e manipulação dessas informações pelo computador recebe o nome de "sistema gerenciador de banco de dados" ou, de modo abreviado, SGBD.

Sistemas gerenciadores

A capacidade dos SGBDs varia bastante: os mais simples não passam de agendas de endereço eletrônicas que acessam um registro ao receberem um parâmetro simples como NOME = ?. Os SGBDs mais avançados contam com suas próprias linguagens de programação incorporadas, manipulando as informações por métodos bastante complexos e permitindo, inclusive, a transferência de campos de dados para outros programas (como folhas de pagamento, planilhas e processadores de texto).

Em resumo, um gerenciador de bancos de dados eficiente permite que o usuário defina o tipo dos dados admitidos em cada campo e, portanto, irá rejeitar uma entrada como esta: NOME = 143326 ou ANO DE ADMISSAO = 1530. Além disso, ele sempre emitirá uma mensagem de erro quando o usuário tentar encontrar registros claramente absurdos do tipo IF NOT (MASCULINO OU FEMININO) AND SALARIO < 0. Ao contrário das linguagens de programação como o BASIC — e de modo análogo às do tipo PASCAL —, os SGBDs exigem uma "tipificação rígida" para que não seja possível a introdução de dados errôneos durante a criação ou modificação dos registros.

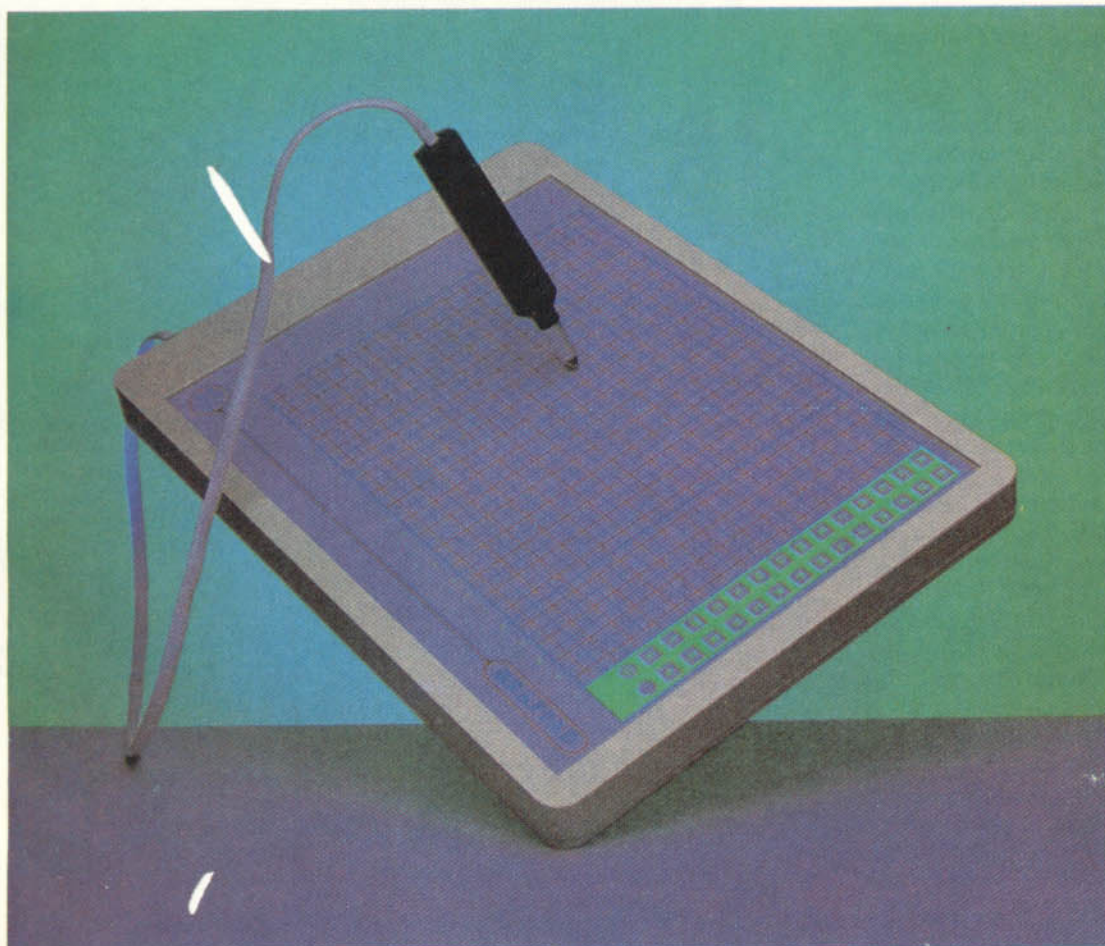
GRAFPAD

Compatível com o TK 90X, o Commodore 64 e o BBC Micro, o tablete digitalizador Grafpad, com seus três pacotes de software, é um equipamento ideal para a produção de desenhos e projetos em micros.

Tabletes digitalizadores estão entre os mais versáteis e úteis periféricos para microcomputadores. Têm importante aplicação como instrumento para desenho, arte a mão livre, projetos de circuitos eletrônicos, cartografia etc. E, além das aplicações diretas em desenho, constituem úteis dispositivos complementares de entrada. A máscara que os recobre pode mostrar todos os recursos disponíveis num programa, em palavras ou símbolos visuais. Tudo o que se tem a fazer é tocar o comando apropriado com a "caneta" que integra o equipamento, e o software processa a opção escolhida.

Tais sistemas eram utilizados em aplicações especializadas, vendidas sobretudo para projetistas e desenhistas. Mas os preços reduziram-se o suficiente para permitir a aquisição por usuários de micros domésticos. O Grafpad é um dos líderes entre os modelos de baixo custo, graças a suas boas características. Pode ser encontrado em versões específicas para o BBC Micro, o Commodore 64 e o Sinclair Spectrum (cujo compatível brasileiro é o TK 90X). A versão aqui ilustrada serve para o BBC.

O equipamento consiste em três elementos: o tablete, uma caneta e o software controlador. O tablete é conectado à saída do bus do micro e a caneta inserida lateralmente. A superfície do tablete apresenta-se dividida em 16 x 20 quadrados e mais uma coluna de comandos (um painel à direita com um conjunto de letras e números). Essa coluna permite o controle de partes do software, sem necessidade de se utilizar o teclado. Uma camada de plástico recobre e protege toda

Idéias gráficas
O Grafpad pode ser usado, com seu próprio software, para criar desenhos; ou, com os programas desenvolvidos pelo usuário, como periférico de entrada.

a superfície do tablete. É possível desenhar máscaras específicas com seus próprios comandos, e nelas traçar reticulados.

No interior do tablete existe um reticulado de 320 x 256 fios, distanciados 1,2 mm. A ponta da caneta é um contato minúsculo. Quando ela toca a cobertura de plástico do tablete, um chip ULA (Uncommitted Logic Array, "matriz lógica não comprometida") varre todos os fios, até detectar, por variação de capacitância, a posição da caneta. Esse rastreamento ocorre 2.000 vezes por segundo, garantindo a rápida localização do ponto tocado. Empunha-se a caneta pela faixa de metal que envolve sua ponta, a fim de ajudar o funcionamento seguro do sistema.

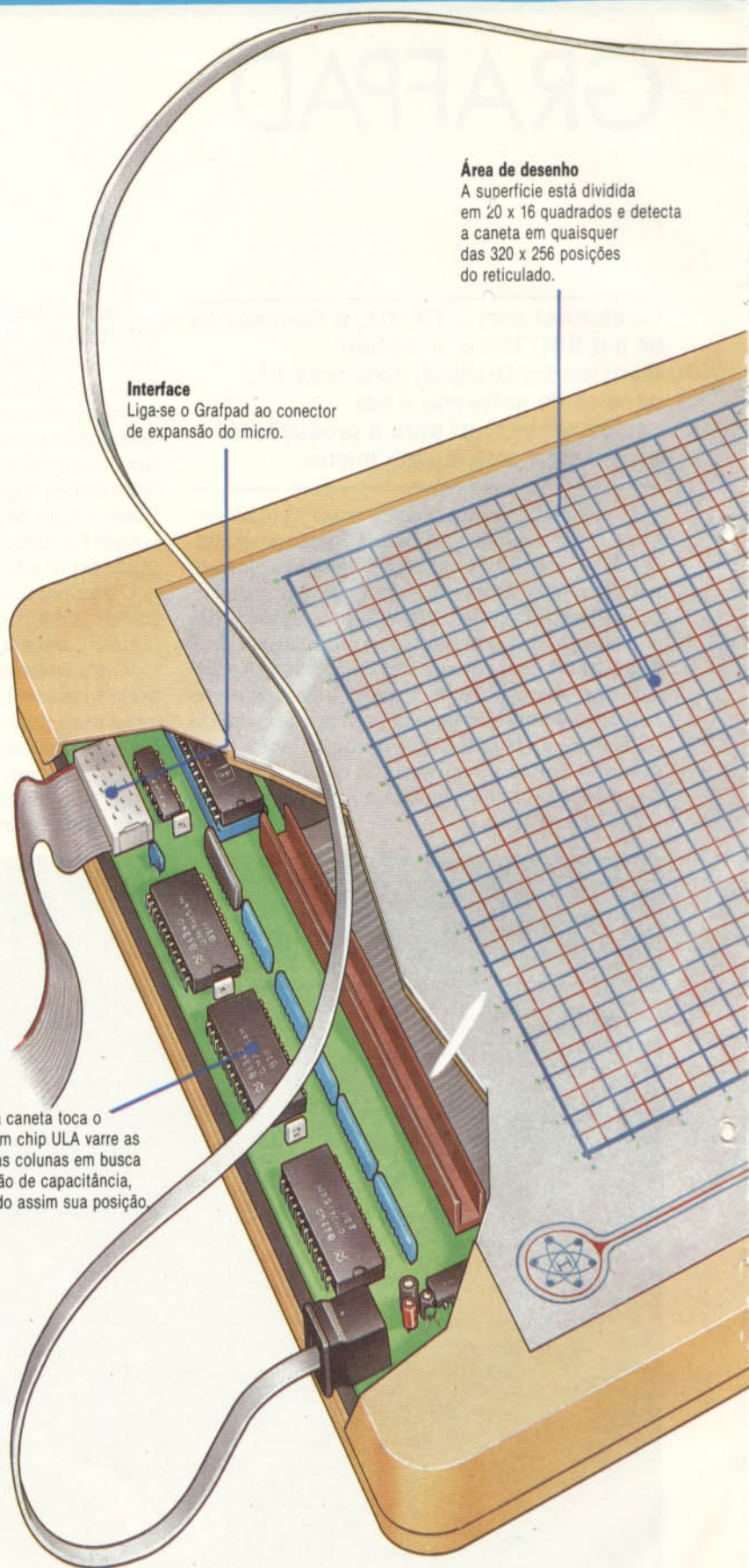
Quando a caneta encosta no tablete, o computador recebe a informação de suas coordenadas. O software determina o efeito exato que isso cria. Pode surgir um cursor em forma de cruz no vídeo, na posição correspondente, ou determinado comando ser acionado. E aqui a economia na construção do Grafpad começa a atrapalhar. Ele só detecta a caneta num reticulado de 320 x 256 posições, o que dificulta o desenho de detalhes muito reduzidos ou uniformes. E, também, o tablete é muito pequeno.

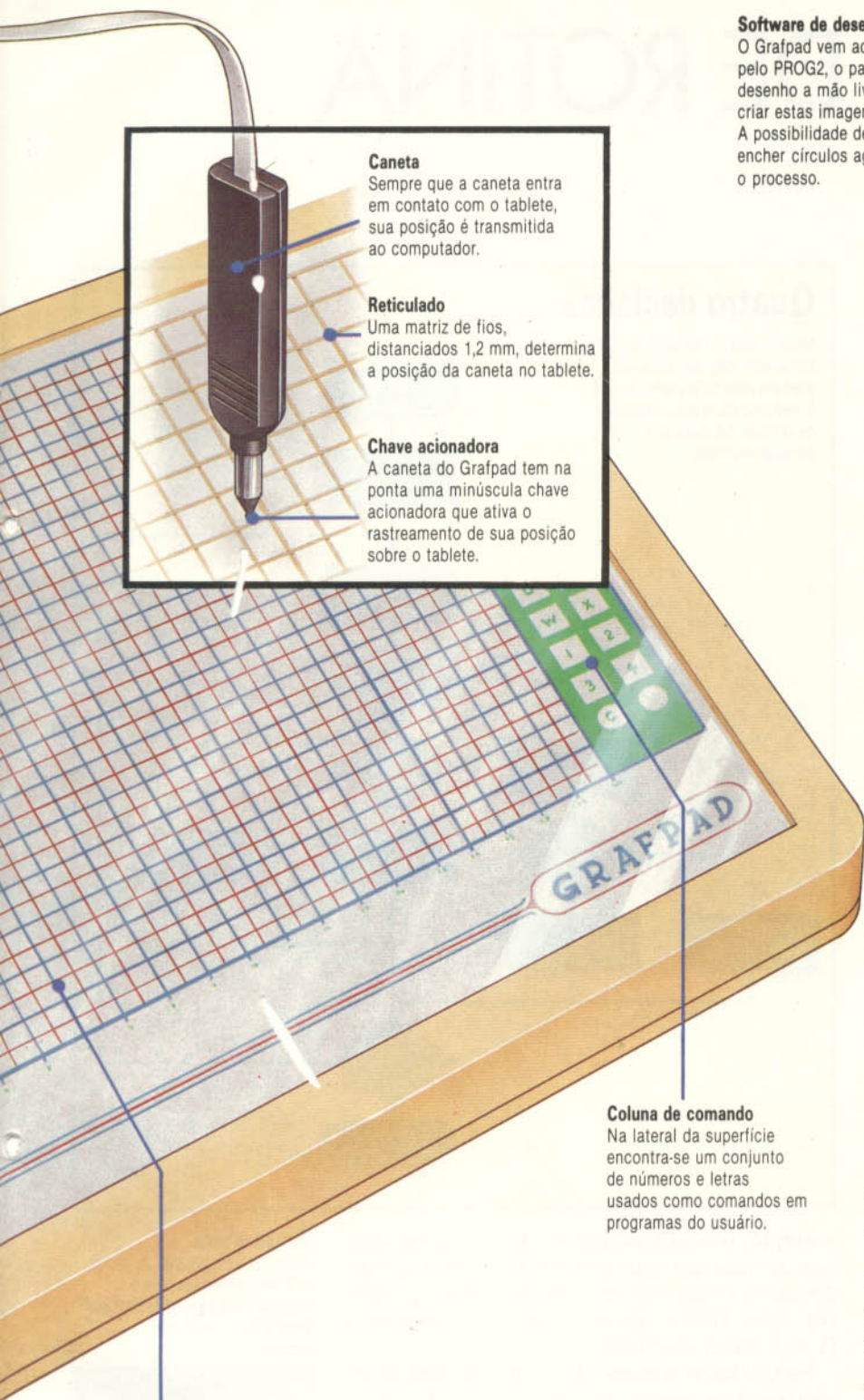
O Grafpad tem três pacotes de software, que vão de simples rotina de demonstração, via um programa de desenho, até um complexo pacote CAD (Computer-Aided Design, "desenho com auxílio de computador"). A rotina simples, tipo conheça-seu-tablete, pode ser incorporada aos programas do usuário (é fornecida em linguagem de máquina e em versão BASIC).

A rotina de desenho — um programa tipo faça-um-esboço — compara-se à maioria dos pacotes de arte disponíveis, mesmo àqueles que não usam tablete. E oferece todas as características básicas: retas, quadriláteros, círculos, triângulos e a mão livre, podendo preencher com cor áreas delimitadas. Faltam, porém, recursos mais sofisticados, como copiar e movimentar partes de um desenho. Nada há aqui que um software direcionado apenas pelo teclado não possa fazer, embora o Grafpad permita a criação de desenhos. A versão para o BBC apresenta apenas quatro cores de cada vez, com a desvantagem de respostas muito lentas.

O programa CAD é uma demonstração de alguns dos princípios envolvidos. Em primeiro lugar, criam-se os caracteres que serão usados nos desenhos. Para a eletrônica, por exemplo, essas formas seriam de componentes transistores, resistores etc. Também se podem criar portas lógicas, peças de mobiliário e mesmo padrões para azulejos. Transferem-se então esses caracteres para a prancheta, onde podem ser repetidos, arranjados e unidos a retas.

Isso é muito parecido com o funcionamento real de um pacote CAD. Mas o software do Grafpad não serve para uso profissional, pois faltam-lhe recursos para tanto. Por exemplo: capacidade de legendar diagramas, girar e graduar em escala os desenhos, ampliar determinada par-

Área de desenho
A superfície está dividida em 20 x 16 quadrados e detecta a caneta em quaisquer das 320 x 256 posições do reticulado.

Interface
Liga-se o Grafpad ao conector de expansão do micro.

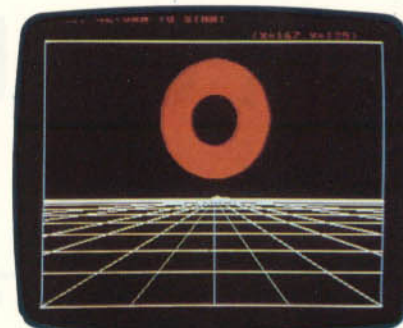
Circuitos
Quando a caneta toca o tablete, um chip ULA varre as linhas e as colunas em busca de variação de capacitância, localizando assim sua posição.

Caneta
Sempre que a caneta entra em contato com o tablete, sua posição é transmitida ao computador.

Reticulado
Uma matriz de fios, distanciados 1,2 mm, determina a posição da caneta no tablete.

Chave acionadora
A caneta do Grafpad tem na ponta uma minúscula chave acionadora que ativa o rastreamento de sua posição sobre o tablete.

Coluna de comando
Na lateral da superfície encontra-se um conjunto de números e letras usados como comandos em programas do usuário.

Cobertura
Um plástico transparente protege a superfície do equipamento, podendo também fixar máscaras para outras aplicações.

Software de desenho
O Grafpad vem acompanhado pelo PROG2, o pacote de desenho a mão livre usado para criar estas imagens.
A possibilidade de traçar e encher círculos agiliza bastante o processo.

te da tela, posicionar com precisão formas pequenas, e assim por diante.

Maior flexibilidade na correção de erros é essencial e, em geral, o programa CAD despreza o uso do Grafpad como periférico de entrada. Apesar da pequena coluna de comandos no tablete, muitos outros precisam ser introduzidos pelo teclado, e a operação, no todo, torna-se muito incômoda.

O Grafpad, em si, mostra-se um periférico versátil e, ainda que limitado quanto a área, resolução e confiabilidade — em função de seu baixo custo —, oferece bom desempenho. O software que acompanha o sistema, no entanto, desaponta — a unidade atrai mais aqueles que querem escrever seus próprios programas. Ainda assim, tabletes como este permitem a exploração de novas possibilidades e, por isso, devem proporcionar impulso considerável no campo de desenhos mais avançados para microcomputadores.

CHECK-UP DE ROTINA

Não há meios de testar, em tempo normal, um programa de complexidade razoável. Mesmo assim, vale a pena criar métodos seqüenciais de teste, a partir da estrutura piramidal do programa.

O BASIC ou qualquer outra linguagem interpretada permite testar as instruções quando são escritas. Pode-se digitar RUN a qualquer momento e ver o que acontece. Na maioria das máquinas, procurar erros resume-se a interromper com BREAK um programa em execução, obter com um PRINT os valores das principais variáveis, verificar esses valores e, então, CONTinuar. Embora erros mais óbvios sejam detectados e corrigidos, esse controle não substitui os testes aplicados quando o programa está acabado.

O teste de validade tem por objetivo garantir que um programa fará exatamente o que se espera dele. Para qualquer conjunto admissível de dados de entrada, ele deve produzir a saída correta; e, para qualquer entrada não permitida, deve executar as ações apropriadas. Aparentemente, pode-se fornecer ao programa uma amostra de todas as entradas admissíveis, mas isso seria inviável para a maioria dos programas. Um que servisse apenas para somar dois números inteiros e imprimir o resultado deveria ser testado para todos os valores possíveis — sem falar nos valores "ilegais", que também precisariam ser testados.

Outra possibilidade seria examinar todos os caminhos lógicos ao longo do programa. Pode-se encontrar cada um deles seguindo uma rota ao longo do diagrama de controle de fluxo (fluxograma), do começo até o fim. Cada ramificação de trajeto leva a caminhos alternativos, e cada loop acrescenta novas opções. A figura 1 mostra um programa simples, um loop contendo várias instruções IF-THEN. Há vários caminhos dentro do corpo do loop, que é executado dez vezes. São cerca de doze linhas de instrução, mas o número de caminhos diferentes chega a 1.398.100, o que descarta essa maneira de testar.

Não funciona nem o teste exaustivo dos dados nem o teste exaustivo da lógica; na verdade, não existem meios de testar um programa de certa complexidade num tempo razoável. Apesar disso, vale a pena criar métodos de teste. Uma suposição razoável é a chamada "equivalência de classe": se a máquina opera corretamente com um dado de determinado tipo, ela irá operar corretamente com todos os dados desse tipo. Por exemplo, quando um trecho de instruções verifica se uma entrada está na faixa de 1 a 100, devem-se testar entradas menores que 1, maiores que 100 e dentro da faixa esperada (1 = < valor = < 100).

Quatro decisões

Mesmo uma construção simples como este loop não pode ser exaustivamente testada, devido à multiplicidade das condições de entrada: há mais de 1 milhão de rotas distintas.

figura 1

Simplifica-se o exame dos caminhos lógicos assinalando os pontos de entrada de cada rotina (em princípio, deveria haver apenas uma entrada para cada rotina) e, dentro das rotinas, cobrindo todos os resultados possíveis de cada ramificação de decisão. Na figura 2, temos uma rotina para ajustar pontos de bônus num jogo que considera os parâmetros de entrada BONUS, NIVEL e ACERTOS. Ela pode ser escrita:

```
6030 IF NIVEL > 2 AND ACERTOS = 10 THEN
     BONUS = BONUS * NIVEL
6040 IF NIVEL = 6 OR BONUS > 2000 THEN
     BONUS = BONUS + 100
```

Testando o teste

Um conjunto completo de dados de teste calculados para o exemplo ilustrado no fluxograma deveria ser semelhante ao seguinte:

ENTRADA			SAÍDA
NIVEL	ACERTOS	BONUS	BONUS
6	10	200	1300
4	10	550	2300
7	10	550	3950
4	10	200	800
7	10	200	1400
1	20	2500	2600
1	20	550	550
6	5	200	300
6	50	200	300
4	5	2500	2600
7	50	2500	2600
4	50	550	550
7	5	550	550

Para cobrir os resultados de cada expressão condicional, precisaríamos considerar as entradas de cada uma que causariam saída "sim" ou "não". Em ambas as decisões, olhamos os efeitos de duas variáveis combinadas por um operador lógico (E e OU). Ou seja, consideramos os valores combinados das variáveis, e não seus valores individuais. Para entender a razão disso, imaginemos o que aconteceria se testássemos os valores 4 e 1 para NIVEL, e 10, 5 e 20 para ACERTOS na primeira decisão. Quando NIVEL=4, os três valores de ACERTOS seriam testados; mas não quando NIVEL=1. Nesse caso, parte de uma decisão teria "mascarado" outra parte. Então, se pudermos testar cada parte separadamente, será melhor simplificar decisões compostas.

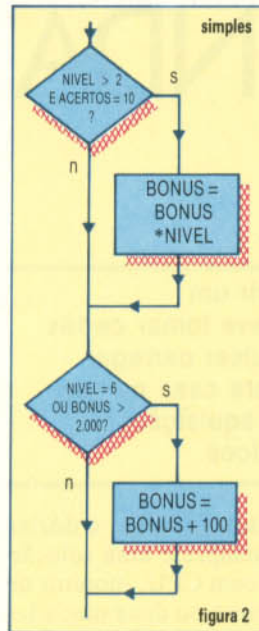
A figura 3 mostra que, com quatro decisões binárias, existem dezesseis (2^4) resultados possíveis; não precisamos cobrir todos eles. Um bom começo é tabelar as condições que produzem resultados sim ou não para cada decisão:

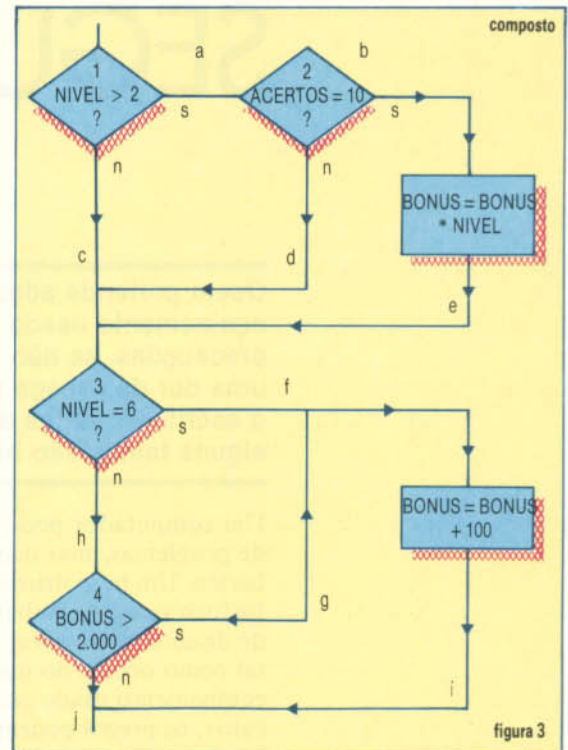
	1	2	3	4
sim	NIVEL>2	ACERTOS=10	NIVEL=6	BONUS>2000
não	NIVEL=2 NIVEL<2	ACERTOS<10 ACERTOS>10	NIVEL<6 NIVEL>6	BONUS=2000 BONUS<2000

A tabela pode então ser usada para derivar valores dos dados representativos do teste. Por exemplo, para o percurso a-d-f-i (figura 3), NIVEL precisa ser maior que 2 e igual a 6, ACERTOS deve ser diferente de 10 e BONUS pode ter qualquer valor porque não está envolvido nas decisões. Os valores NIVEL=6, ACERTOS=20 e BONUS=150 seguiriam esse caminho — como fariam tantos outros. O percurso a-b-e-h-j poderia ser testado com NIVEL=4, ACERTOS=10 e BONUS=600 (falamos do valor de *entrada* de BONUS, que mais tarde pode ser multiplicado pelo valor de NIVEL).

Devem-se calcular previamente os resultados a serem obtidos com cada conjunto de dados, para comparação aos produzidos pelo teste. Os dados de entrada meramente testarão se o programa "roda". Para testar se ele faz o que deve, a saída precisa ser calculada de antemão — como no exemplo *à esquerda,* para NIVEL, ACERTOS e BONUS.

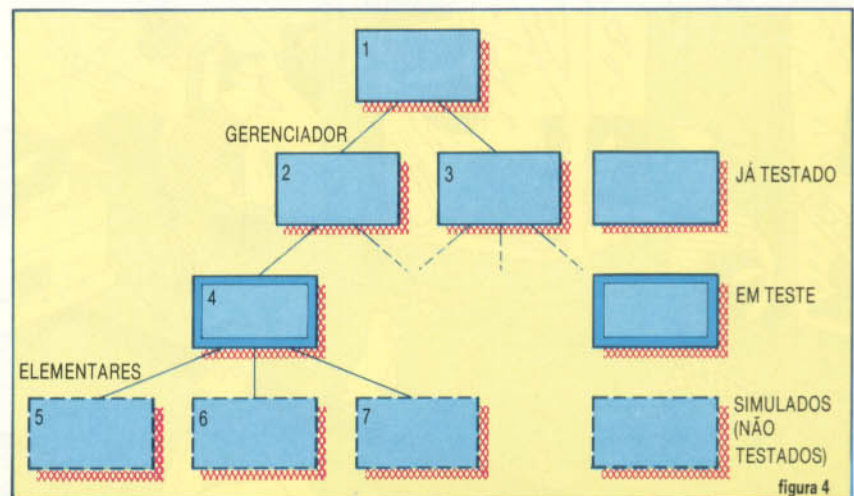
Equipados com um método para "exercitar" nosso software, precisamos agora de um método para abordar programas de maior extensão. Nesse ponto, percebe-se outra vantagem da programação estruturada: programas escritos como uma coleção de módulos independentes dispostos numa hierarquia permitem testar cada módulo individualmente. Podemos começar com o módulo prioritário (o do topo da hierarquia) e "descer" a pirâmide. Só testamos um módulo quando todos acima dele tiverem sido testados, e usamos módulos já testados para fornecer dados aos inferiores.

A menos que seja o primeiro, o módulo a ser testado terá acima dele um módulo gerenciador

composto
figura 3

Decompondo a decisão
Nomeie as ligações e simplifique decisões compostas; isso facilita os testes.

plenamente testado. Os inferiores a ele, que poderíamos chamar de "elementares", ainda não foram testados, não sendo confiáveis.

Simulam-se os elementares por curtos trechos de programa que simplesmente produzem dados apropriados quando utilizados pelo módulo que está sendo testado. Às vezes, chama-se esse arranjo de "couraça de teste" — uma armação em que rotinas modulares podem ser colocadas para teste. A figura 4 corresponde a um esquema desse princípio. Os módulos 1, 2 e 3 foram testados, o módulo 4 está em teste e 5, 6 e 7 são simulados.

Os testes, constituindo importante parte do ciclo de vida de um programa, precisam ser bem documentados. Vale a pena manter registros dos dados de teste derivados de uma rotina. Se surgirem problemas, não haverá necessidade de repeti-los, podendo-se verificar, também, por que razão se mostraram inadequados.

Teste do topo à base
Simplifica-se o teste pela abordagem do topo à base, uma vez que cada módulo é testado como se escreve, tanto isolado quanto associado a outros já testados.
O comportamento de módulos ainda não escritos pode ser simulado criando-se rotinas que produzem, em suas saídas, dados gerados artificialmente, mas coerentes com os previstos.

SEGUNDA MÃO

Quem pretende adquirir um equipamento usado deve tomar certas precauções, se não quiser carregar uma dor de cabeça para casa ou para o escritório. Antes da aquisição, alguns testes são básicos.

Um computador pode resolver dúzias e dúzias de problemas, mas nem sempre é uma solução barata. Um bom sistema, com CPU, monitor de fósforo verde ou âmbar, uma ou duas unidades de disco e impressora, custa caro. No entanto, tal como ocorre no mercado de automóveis, o equipamento usado sai mais barato. Em certos casos, os preços podem chegar à metade do cobrado por um computador novo.

Mas você precisa saber o que está comprando. Antes de mais nada, é necessário determinar exatamente para que o micro está sendo comprado — jogos, contabilidade, processamento de texto, cálculos financeiros, folhas de pagamento, faturamento, simulação, controle de processos etc.

Em teoria, qualquer equipamento pode fazer tudo isso (uma coisa de cada vez, no mínimo), mas não compre nada até ter certeza de que existe software apropriado para as tarefas que o micro vai executar.

Nessa etapa, a situação é a mesma de quem compra um micro novo. Convém conhecer o software, experimentá-lo, verificar se ele atende às necessidades da tarefa a ser executada e só então decidir que combinação máquina/programa deve ser escolhida.

Contudo, tal decisão precisa levar em conta outros fatores, como, por exemplo, a adaptabilidade da máquina. Procure saber:

- Se ela tem até 32 Kbytes de memória, há no mercado expansão para 64 Kbytes? (Os programas mais eficientes requerem esta quantidade.)
- A configuração mínima já inclui alguma interface para periféricos?
- Há no mercado interfaces para as aplicações desejadas?
- É grande a variedade de software para essa máquina?
- Quanto custa fazê-la comunicar-se com um

serviço público de transmissão de dados como o Videotexto?

Depois de avaliados esses aspectos, pode-se decidir com segurança pelo menos o tipo de máquina que executará o trabalho pretendido.

A etapa seguinte consiste em procurar o equipamento, e aí o caminho toma duas direções: pode-se buscá-lo nas lojas ou nos classificados de grandes jornais, onde pessoas, pelas mais diversas razões, anunciam equipamentos usados para vender. O preço é questão muito importante, mas não a única. Exemplo: um micro usado vendido numa loja pode ser mais caro do que o comprado de um particular; em contrapartida, a loja em geral oferece alguma garantia (trinta dias, em média), e sempre se pode conseguir software grátis.

Chega, então, o momento de avaliar a máquina para saber se ela está em boas condições. A primeira coisa que se faz é pedir ao dono que ligue o micro — isso pode evitar uma porção de surpresas. Se o equipamento emprega fitas cassete, começa-se o teste carregando um programa na memória do micro. Se não houver, melhor ainda: digita-se um, para testar a gravação e o carregamento. Ele pode ser curto, como o que segue, para equipamentos compatíveis com o TRS-80, facilmente adaptável para outras linhas de micros:

```
10 CLS
20 FOR I=0 TO 255
30 PRINT (0+1), CHR$(I);
40 NEXT I
50 GOTO 50
```

Isso vai exibir todos os caracteres disponíveis em BASIC e servirá também como teste inicial do teclado. Deve-se carregar e rodar o programa mais de uma vez, para ter certeza de que a lógica do micro está em ordem. Por fim, verifica-se o tamanho da memória com um comando do tipo PRINT MEM. Como esse comando pode variar de um BASIC para outro, consulte antes o manual da linguagem. Um micro com 64 Kbytes de RAM deve responder com pelo menos 48 Kbytes livres; se estiver com problemas, poderá dar uma resposta absurda como, por exemplo, 722 bytes livres.

O teclado constitui outro item a ser testado cuidadosamente. O ideal é digitar uma série de instruções, um texto ou experimentar todas as teclas, verificando se elas não ficam presas. E, em caso afirmativo, se repetem caracteres, defeito que às vezes compromete o teclado inteiro. Depois de todos esses testes, passa-se à avaliação do monitor de vídeo. Em dez minutos de funcionamento, ele evidenciará qualquer problema que porventura tenha.

Esse periférico não deve apresentar nenhum ruído (no máximo zumbido de bobinas); os caracteres que aparecem no alto e no pé da tela devem ter o mesmo tamanho; e a imagem precisa

estar centrada — aspecto que se verifica com este programa:

```
10 CLS
20 FOR I=0 TO 1023
30 POKE (15360 I),191
40 NEXT
50 GOTO 50
```

Se o quadro gerado se apresenta descentrado, bastarão alguns ajustes. É o máximo de problemas que se pode aceitar de um monitor.

Para teste da impressora, usa-se o mesmo programa gerador de caracteres na tela, empregando-se LPRINT no lugar de PRINT. Será suficiente para verificar se todas as trilhas de comunicação micro—interface—impressora encontram-se perfeitas, e também se a própria impressora funciona bem.

No caso das unidades de disco, há dispositivos que apresentam problemas com alguma freqüência, por causa de sujeira. Assim, podem não estar funcionando bem apenas por estarem sujos. Um bom teste é colocar o sistema operacional e executar alguns programas gravados em linguagem de máquina (como o próprio interpretador BASIC) e formatar um ou dois disquetes. Se o sistema estiver em ordem, a formatação deverá se concluir com êxito; havendo problema, tenta-se de novo, com um disquete virgem. Se ainda ocorrerem erros durante a formatação, será melhor procurar outro equipamento.

LINGÜÍSTICA ESTRUTURAL

A diferença entre procedimentos e funções, assim como os novos conceitos de "âmbito" e "transferência de parâmetro", ajudará a estruturar de modo mais eficiente os programas em PASCAL.

Finalizando a investigação das estruturas de dados no PASCAL, examinaremos alguns métodos de estruturação de programas. Nesta fase, definiremos novos procedimentos e funções que complementam os fornecidos pela linguagem, como o procedimento WRITE e a função SQRT. Contudo, antes disso, será necessário retomar a distinção entre procedimento e função. Ao dizermos, por exemplo,

```
WRITE ('ALO!')
```

esperamos que o string de caracteres fornecido como parâmetro único apareça no arquivo de saída padrão. Ou seja, WRITE identifica um subprograma responsável pelo envio dos dados, sob a forma de um fluxo de caracteres, para o dispositivo de saída. Mas qual seria o resultado da seguinte "instrução"?

```
SQRT (256)
```

Nenhum, além de gerar uma mensagem de erro, visto não ser uma instrução legítima. Por outro lado,

```
WRITE (SQRT (256))
```

não apenas é válida, como também apresenta na tela algo do tipo 1,60000E + 01. A instrução WRITE identifica um processo que envia dados para a saída, sendo um procedimento designado por seu próprio nome. Um outro procedimento, PAGE (F), serve para iniciar uma nova página no arquivo de texto F.

No caso do identificador SQRT, sempre atribuímos a ele um único "argumento" numérico para que forneça um resultado — a raiz quadrada do argumento. Desse modo, a primeira distinção entre procedimentos e funções é a de que não existe um comando função. Assim como 16 isoladamente nada significa (trata-se apenas de um valor), expressões isoladas como SQRT (X/3) ou ODD (N) também não possuem nenhum significado. Os identificadores de função comportam-se como variáveis não inicializadas, que são calculadas toda vez que seus nomes surgem numa instrução. Calcula-se o valor único resultante a partir do(s) valor(es) do(s) argumento(s) relativo(s) à função — ou seja, o resultado é uma função do(s) argumento(s).

Os procedimentos, ao contrário, não fornecem valores e, portanto, não podem ser usados em expressões. A instrução

```
N := WRITELN
```

é claramente ilegítima, do mesmo modo que LET N = PRINT o seria em linguagem BASIC.

A elaboração de programas extensos fica bastante facilitada, em PASCAL, pela possibilidade de escolhermos os nomes de procedimentos e funções, bem como de controlarmos o acesso a eles e sua vinculação mútua. Considere o seguinte programa:

```
PROGRAM INCOMPLETO (INPUT, OUTPUT,
  ARQUIVO); {DECLARACOES..}
BEGIN
  OPEN (ARQUIVO);
  WHILE NOT EOF (ARQUIVO) DO
    BEGIN
      READ (ARQUIVO,ITEM);
      PROCESSE (ITEM);
    END;
END.
```

Embora você ainda não saiba como manipular arquivos, não será difícil compreender o que faz esse programa. Na parte da declaração há um procedimento para localizar um arquivo e abri-lo para leitura (OPEN), e um outro para manipular cada item de dados (PROCESSE).

O programa também utiliza dois identificadores predefinidos do PASCAL. O procedimento READ (utilizado até agora para ler entradas de texto) serve para a leitura de quaisquer dados, estruturados ou não, provenientes de um arquivo.

A função booleana EOF (End of File, "fim de arquivo") fornece um valor "verdadeiro" quando o último registro do arquivo é lido. Para arquivos de texto, a função EOLN (End of Line, "fim de linha") resultará "verdadeira" quando o último caractere da linha for lido.

A diferença fundamental entre programas, procedimentos e funções é pequena — afinal, todos são módulos com suas próprias descrições de dados locais e seus núcleos de instruções. A única variação sintática diz respeito ao cabeçalho, um programa assim composto: pela palavra reservada PROGRAM; por um identificador, escolhido pelo usuário, que designa o nome do programa; e por uma lista de parâmetros que indica os arquivos utilizados pelo programa.

No cabeçalho de um procedimento, a palavra reservada PROCEDURE substitui PROGRAM, ampliando a "lista de parâmetros formais" de modo a incluir identificadores dos tipos de parâmetros. Assim, por exemplo,

```
PROCEDURE LINHAS (NUMLINHAS : INTEGER);
  VAR
    N : INTEGER;
  BEGIN
    FOR N := 1 TO NUMLINHAS DO
      WRITELN
END;
```

Ao END do procedimento segue-se ponto-e-vírgula, e não ponto-final, como no fim do programa. NUMLINHAS, o identificador de parâmetros formais, recebe o valor efetivo na instrução de chamadas:

```
LINHAS (5)
```

Esse procedimento trivial mostra-se muito útil quando queremos deixar várias linhas em branco (cinco, no caso) para facilitar a leitura, evitando seqüências contínuas de instruções WRITELN separadas por sinais de ponto-e-vírgula.

Âmbito

O exemplo precedente também serve como ilustração da confiabilidade do PASCAL. A variável FOR, para controle de loop, sempre é declarada como uma variável local estrita. Não se diz, portanto,

```
FOR NUMLINHAS := 1 TO NUMLINHAS DO
```

Embora isso seja aceitável num programa principal, a confiabilidade do loop não estará garantida enquanto o controlador do loop for não local, ou "relativamente global". Certamente você já passou horas depurando um programa em BASIC contendo uma instrução como

```
300 FOR N=1 TO T
400 GOSUB 2000
500 NEXT N
```

para descobrir, no final, que uma rotina — talvez empregada de modo indireto e condicional — usava a variável N com algum outro objetivo. O PASCAL não permite, em hipótese alguma, essa ambigüidade, possibilitando assim uma grande economia de tempo na elaboração dos programas. Qualquer identificador declarado no interior de um bloco opera numa "região" definida pelos limites do bloco. No entanto, esse "âmbito" (a parte do bloco a partir da qual o identificador é acessível) estende-se apenas de sua declaração até o final do bloco, e poderia ser limitado ainda mais por uma redefinição. No exemplo anterior, o N referido no procedimento LINHAS é um valor integer local, cancelando, a cada execução do procedimento, qualquer outra declaração relativamente global.

No esquema do programa principal (AMBITO), mostrado no quadro *acima,* a região das variáveis globais N e X corresponde a todo o programa. O âmbito de X vai desde sua declaração até o final do programa, excetuando-se o interior do procedimento A. O motivo está na existência de um outro X (SET OF CHAR) definido como identificador de tipo, cuja região é o bloco A. De maneira análoga, no interior de B, N refere-se ao parâmetro formal de B, e não à variável global.

Embora o âmbito tanto de A como de B corresponda a todo o programa, o de B só começa a partir de seu ponto de definição. Assim, enquanto for possível utilizar A no procedimento B, este será inacessível para A. Isso reforça a lógica rigorosa do PASCAL, impossibilitando a execução de qualquer programa até que tenha sido escrito. Por isso, as definições e declarações vêm, obrigatoriamente, na ordem CONST, TYPE, VAR, seguidas de procedimentos e funções, de acordo com os requisitos estruturais do programa. Muitos compiladores do PASCAL são de "passagem única" (lêem o código-fonte uma única vez), impossível sem uma rígida ordenação lógica.

O esforço adicional para declarar os dados locais em qualquer procedimento é amplamente recompensado. Uma das vantagens é a obtenção da modularidade por meio da transferência, para procedimentos, de todos os valores de dados sob a forma de parâmetros. Embora existam programas em PASCAL cujos procedimentos não apresentam nenhuma lista de parâmetros (todos os dados são acessados globalmente), essa não é uma boa prática de programação.

Observe que, assim como qualquer referência a X no interior de A implica o tipo X, o uso de

Âmbito abrangente

```
PROGRAM AMBITO;
VAR
    N : INTEGER;
    X : REAL;
PROCEDURE A (Y : REAL);
  TYPE
    X=SET OF CHAR;
  {ETC.}
PROCEDURE B (N : INTEGER);
  {ETC.}
BEGIN {AMBITO PRINCIPAL}
  ...
  A (SUCC (N)/3);
  B (N);
  A (X);
    {ETC.}
END.
```

No programa AMBITO, aqui delineado, as regiões e os âmbitos das diferentes variáveis e procedimentos são os seguintes:

Proc/Var	Região	Âmbito
A	Principal	A, B, principal
B	Principal	B, principal
N (global)	Principal	A, principal
X (real)	Principal	B, principal
X (no proc A)	A	A
N (no proc B)	B	B

N no interior de B refere-se a seu parâmetro local, deixando o valor integer global N temporariamente inacessível. A chamada B transfere o valor do global N como um parâmetro efetivo, designado por acaso com o mesmo nome em B, mas indicando uma variável diferente. Disso resultam vários pontos diferentes:

- Uma cópia local do valor transferido é feita na entrada de um bloco.
- Qualquer alteração desse ''parâmetro de valor'' no interior de seu bloco não altera o parâmetro efetivo transferido pela chamada.
- O valor transferido pode ser qualquer expressão do tipo adequado.

Qualquer instrução em um procedimento relaciona os parâmetros efetivos da chamada, os quais correspondem (quanto ao número, posição e tipo) à lista de parâmetros formais no cabeçalho. O mecanismo para transferência do valor de parâmetro pode ser :onsiderado como uma inicialização do identificador de parâmetros formais para qualquer valor especificado na instrução de chamada. Assim,

```
LINHAS (SUCC (N + A) DIV 2)
```

implica a seguinte instrução ıtributiva na entrada para LINHAS:

```
NUMLINHAS := SUCC (N + A) DIV 2
```

De uma base para outra

O programa CONVBASE exemplifica o uso de procedimentos com parâmetros. Você escolhe qualquer base entre 2 (binária) e 16 (hexadecimal), e os números decimais introduzidos serão apresentados na notação apropriada. Por exemplo, 32767 seria representado por 7FFF em hexadecimal, 77777 em octal ou 111111111111111 em binário. Se você quiser operar em sentido inverso, use o programa como modelo para, tomando um número em qualquer base (de 2 a 16), obter o resultado sob forma decimal. Que tal incorporar essa modificação como um procedimento do programa abaixo? Outro exercício seria imaginar um modo de fazer com que o programa opere com representações negativas. Para isso, devemos converter o número no seu complemento a 2. Ao nível de máquina, obtém-se isso pela negação do número e adição de uma unidade.

```
PROGRAM CONVBASE (INPUT, OUTPUT);
        {CONVERTE NUMEROS DECIMAIS PARA QUALQUER
          BASE DE BINARIO A HEXADECIMAL}
CONST
        COLUNAS     = 79; {PARA VIDEO/IMPRESSORA}

TYPE
        BYTE        = 0 .. 255;
        CARDINAL    = 0 .. MAXINT;

VAR
        NUMERO,
        BASE        : INTEGER;
        TROCAR,
        LEGAL       : BOOLEAN;

        {11111111111111111111111111111111}

PROCEDURE IMPREDIGITO (DIGITO : BYTE);
    {VALOR DE LISTA [CONTADOR] PASSADO A DIGITO}
  BEGIN
    IF DIGITO IN [0 .. 9]
      THEN
        WRITE (DIGITO : 1)
      ELSE          {REPRESENTAR COMO A .. F}
        CASE DIGITO OF
          10        : WRITE ('A');
          11        : WRITE ('B');
          12        : WRITE ('C');
          13        : WRITE ('D');
          14        : WRITE ('E');
          15        : WRITE ('F');
        END;        {CASE}
  END; {IMPREDIGITO}
        {11111111111111111111111111111111}

PROCEDURE    IMPRIME    (N          : CARDINAL;
                         BASE       : BYTE);
  CONST {TODOS ESTES DADOS SAO LOC  IS PARA IMPRIME}
        MAXDIGITOS = 32;
  TYPE
        LIMITES     = 1 .. MAXDIGITOS;
  VAR
        LISTA       : ARRAY [LIMITES] OF E  TE;
        INDICE      : LIMITES;
        CONTADOR    : BYTE;
  BEGIN
    CONTADOR := 0;

    REPEAT
        CONTADOR := SUCC (CONTADOR);
        LISTA [CONTADOR] := N MOD BASE;
        N := N DIV BASE

    UNTIL N = 0;
        {IMPRIME COMECANDO PELO MAIS SIGNIFICATIVO}
    FOR CONTADOR := CONTADOR DOWNTO 1 DO
        IMPREDIGITO (LISTA [CONTADOR]);
    END; {IMPRIME}
        {11111111111111111111111111111111}
BEGIN {CONVBASE — PROGRAMA PRINCIPAL}

    REPEAT
      WRITELN ('ESCOLHA UMA BA  E : ');
      WRITELN ('        2 .. 16         ' : COLUNAS);
      WRITELN ('(QUALQUER O   RA ENCERRA)' : COLUNAS);
      WRITE ('BASE?');
      READ (BASE);
      LEGAL := BASE IN [2 .. 16];

      IF LEGAL THEN
        BEGIN
          WRITE ('NUMERO (0 TROCAR BASE)?');
          READ (NUMERO);
          TROCAR := NUMERO <= 0;

          WH  E NOT TROCAR DO
            BE  N
              WR  E (NUMERO : COLUNAS DIV 2,
                'TO BASE', BASE : 1, 'E');
              IMPRIME (NUMERO, BASE);
              WRITELN;
              WRITE ('NUMERO?');
              READ (NUMERO);
              TROCAR := NUMERO <= 0;
            END
        END

    UNTIL NOT LEGAL

END.
```

SUMÁRIO

VOLUME 1

APLICAÇÕES

CHIPS & BYTES

CIÊNCIA DA COMPUTAÇÃO

LINGUAGENS

OFICINA

PERFIL

PERIFÉRICOS

PROJETOS DE PROGRAMAS

RAIO X

SOFTWARE

TÉCNICAS DE PROGRAMAÇÃO